A Perspective on IEEE Time-Sensitive Networking for Industrial Communication and Automation Systems

This paper addresses the time-sensitive networking and provides an overview of its introduction in the industrial communication scenario, along with an analysis of the future directions of research.

By Lucia Lo Bello and Wilfried Steiner

ABSTRACT | Networks are a core element of many industrial and automation systems at present. Often these networks transport time- and safety-critical messages that control physical processes. Thus, timely and guaranteed delivery is an essential property of networks for such critical systems. Over the past decade, a variety of network solutions have evolved to satisfy said properties. However, these solutions are largely incompatible with each other and many system architects are forced to deploy different solutions in parallel due to their different capabilities. IEEE 802.1 Time-Sensitive Networking (TSN) is a standardization group that enhances IEEE networking standards, most prominently Ethernet-based networks, with said properties and has the unique potential to evolve as a cross-industry mainstream networking technology. In this survey paper, we give an overview of TSN in industrial communication and automation systems and discuss specific TSN standards and projects in detail as well as their applicability to various industries.

KEYWORDS | Automotive networks; fault tolerance; IEEE 8021; industrial automation; real-time networks; time-sensitive networking (TSN)

Manuscript received January 25, 2019; revised March 7, 2019; accepted March 12, 2019. Date of publication April 23, 2019; date of current version May 28, 2019. This work was supported in part by the University of Catania through the CHANGE Project. *(Corresponding author: Lucia Lo Bello.)*

L. Lo Bello is with the Department of Electrical, Electronic and Computer Engineering, University of Catania, 95125 Catania, Italy (e-mail: lucia.lobello@unict.it).

W. Steiner is with TTTech Computertechnik AG, 1040 Vienna, Austria (e-mail: wilfried.steiner@tttech.com).

Digital Object Identifier 10.1109/JPROC.2019.2905334

I. INTRODUCTION

In many contexts, such as industrial and automotive networks, critical traffic flows generated by applications that require bounded low latency and low jitter share the communication channel with flows originating from applications with less severe timing constraints in order to improve efficiency and to reduce cost. Under these conditions, it is imperative to guarantee the timing behavior of critical traffic and provide temporal isolation from noncritical communication. However, timing is not the only critical aspect to be considered, like reliability, fault tolerance, and security are also other crucial requirements for these applications. The standardization efforts within the IEEE 802.1 Time-Sensitive Networking (TSN) task group,[1] which is part of the IEEE 802.1 Working Group, go in this direction. TSN finds its roots in the IEEE 802.1 Audio Video Bridging (AVB) set of standards. AVB provides several features, such as the specification for low-latency traffic flows in IEEE 802.1 networks, bandwidth reservation to set aside a certain amount of guaranteed bandwidth across a portion of the network for handling this traffic, and, last but not least, protocols to manage the network time in order to support synchronized operations (i.e., A/V playback). However, while AVB significantly improved the real-time performance of IEEE 802.1 networks, the defined features are still not enough to support broad classes of time-sensitive or mission-critical traffic flows in the automotive and industrial automation industries. Novel mechanisms and augmented capabilities

[1] https://1.ieee802.org/tsn/

are needed in IEEE 802.1 bridges and end stations to guarantee worst case end-to-end (E2E) latency, high reliability, bandwidth isolation, and zero congestion loss. TSN standardizes these functionalities to close the remaining performance gaps. Teener *et al.* [1] detail the AVB mechanisms. In a recent survey paper, Nasrallah *et al.* [2] give a general overview of current techniques to achieve ultralow latency communication (including TSN). Jiang *et al.* [3] address low-latency communication in mobile and fixed networks, whereas Seno *et al.* [4] show that the performance figures of industrial communication systems measured in real applications can be often worse than those expected due to the combination of multiple sources of nondeterminism. The recent survey in [5] provides an overview of the next generation of automotive systems, with a focus on TSN among the networked technologies. TSN has also been the topic of recent works written by some members of the IEEE TSN Task Group, such as the recent editorial in [6] and the two overview papers by Finn [7] and Messenger [8], respectively.

Complementary to the previous work, this paper addresses in detail and critically reviews both the TSN standards relevant to industrial communication and automation systems published so far and some ongoing projects, with the purpose to highlight how and to what extent these standardization efforts empower Ethernet bridges to support the new requirements raised by current and future industrial use cases (in this paper, we use the terms "bridges" and "switches" synonymously). This paper first introduces basic IEEE 802.1 networking concepts in Section II and then continues with an overview of the main IEEE TSN standards in Section III. This paper then focuses on core TSN standards for real-time and fault-tolerant communication. In Section IV, we discuss the clock synchronization capabilities of TSN (IEEE 802.1AS/ASrev), real-time enhancements, with a focus on the TSN approaches to support scheduled traffic (IEEE 802.1Qbv-2015), frame preemption of low-critical frames by high-critical frames (IEEE 802.1Qbu-2016), traffic filtering and policing (IEEE P802.1Qci-2017), redundant frame transmission (IEEE 802.1CB-2017), as well as configuration aspects (IEEE 802.1Qcc). This paper also discusses how these novel features make TSN an enabler for several cutting-edge technologies in areas, such as Industry 4.0, autonomous driving, and others in Section V. A focus is given to industrial automation and automotive applications since these two domains are currently driving the TSN developments. In Section VI, we discuss advantages and disadvantages of the network solutions based on TSN and give an outlook to relevant research directions. This paper concludes in Section VII.

II. IEEE 802.1 AND ETHERNET BASICS

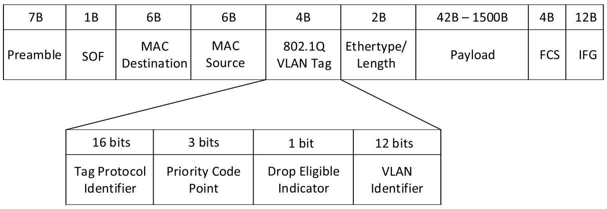
In this section, we discuss basic networking concepts of the IEEE 802.1 and will focus on the realization of IEEE 802.1 standards in switched Ethernet (IEEE 802.3).

A. Switched Ethernet Network Operation

A switched Ethernet network comprises end stations and bridges (in Ethernet terminology bridges are frequently also referred to as switches). End stations will typically be producers and consumers of application data, while bridges will mostly relay such application data between end stations. End stations exchange data between each other by encapsulating said data in Ethernet frames and transmitting them to bridges, which will either forward the frames to one or many final receiving end stations or to other bridges in the network closer to the end stations. The layout of a typical Ethernet frame is depicted in Fig. 1.

Some relevant fields in an Ethernet frame are the address fields of the sender (MAC source) and the receiver(s) (MAC destination) as well as the virtual local area network (VLAN) tag. The VLAN tag itself consists of the following fields: the tag protocol identifier (always set to 0 × 8100 to identify the frame as VLAN-tagged frame), the priority code point (assigning the frame one of eight priorities), the drop eligible indicator, and the VLAN identifier (assigning the frame to a particular VLAN).

The forwarding tasks of a bridge can be roughly classified into: traffic switching, traffic shaping, and traffic policing. Since a bridge will typically have multiple ports say, for example, four ports, traffic switching is the task of the bridge to determine, for a frame received on one of the ports, the reception port, and the set of transmission ports to which the frame shall be forwarded. Ethernet transmissions may be unicast, multicast, or broadcast. Hence, the set of transmission ports can potentially be any number of ports (with the exception that a frame will usually not be transmitted to its reception port). While there are many Ethernet switches on the market, only some provide a guarantee of linespeed traffic switching, i.e., the ability to forward arbitrary incoming traffic patterns without frame loss (under the assumption that the capacity of the transmission port is not exceeded). An example of such an arbitrary incoming traffic pattern is: each port receives minimum-sized Ethernet frames back to back, all frames are unicast, and all frames from a port i need to be forwarded to a port $i + 1$ (and the last port forward to the first port). To guarantee linespeed operation in such a scenario, the traffic switching function must complete within a timespan that is well below the reception time of a minimum-sized Ethernet frame, which is challenging



7B	1B	6B	6B	4B	2B	42B – 1500B	4B	12B
Preamble	SOF	MAC Destination	MAC Source	802.1Q VLAN Tag	Ethertype/ Length	Payload	FCS	IFG

16 bits	3 bits	1 bit	12 bits
Tag Protocol Identifier	Priority Code Point	Drop Eligible Indicator	VLAN Identifier

Fig. 1. ***Layout of an Ethernet frame with VLAN tag.***

to the design in hardware and, with growing linespeeds, almost impossible without hardware switching fabrics.

Traffic shaping describes scheduling mechanisms at the transmission ports. It will be frequently the case that multiple frames concurrently become ready for transmission at a transmission port, for example, when many transmitting end stations send to a common receiving end station or at ports that connect two bridges to each other (so-called multihop links). We summarize the scheduling mechanisms that decide which of the potentially many frames is selected for transmissions as traffic shaping. Traffic shaping to enhance real-time performance has been the key focus of TSN.

When the network is deployed in critical systems, we may also implement protection mechanisms to shield the network from broken equipment and other failures. We call such protection mechanisms traffic policing. Such policing can be both: checks on the frame content and checks on the frame timing or frequency. Traffic policing has been another focus of TSN.

B. IEEE 802.1 Standard Formalism

Although the basic principle of communication in a switched Ethernet network is explained in a couple of paragraphs (like the ones mentioned above), the development of an IEEE 802.1-conformant bridge is far from trivial. Likewise, the necessary modifications and extensions to realize the TSN functionality for real-time and robust communication were quite challenging. Therefore, we briefly discuss the formalism of IEEE 802.1 next.

The central reference document of IEEE 802.1 is the IEEE 802.1Q standard [9], named bridges and bridged networks; for simplicity, we will use the abbreviation *Q* to refer to the standard in this paper. Many standardization projects of various task groups, including TSN, modify and extend Q and, as a result, this standard is just below 2000 pages at the time of this writing. Naturally, we may only present key concepts and strong simplifications in this paper. We provide links to relevant clauses of the standard for further studies to the interested reader.

As the title implies, Q defines bridge functionality as well as the operation of one or many bridges in a bridged network. Clause 5 of Q is the so-called conformance clause, which defines mandatory functionality as well as optional extensions that shall/should/may be implemented by a device, such that it is rightfully entitled to be called a bridge. More precisely, in the course of TSN, we are interested in a special type of bridge, the VLAN bridge as defined under clause 5.4 of Q. For our considerations of TSN, we further elaborate on items d) and e) of the said clause that refers to clause 8, which defines mandatory frame relaying and filtering functionality.

Q defines the forwarding tasks of frames within a bridge in clause 8. An overview of this forwarding functionality is shown in Fig. 2 (numbers in brackets refer to the respective clauses in Q).

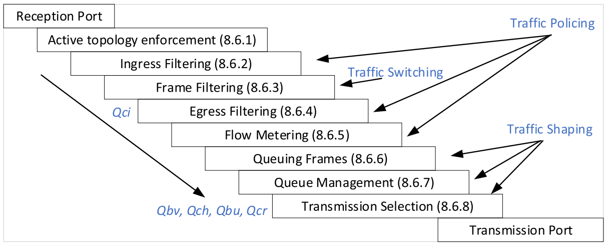


Fig. 2. ***Overview of the frame forwarding process in a bridge.***

Traffic switching is the focus of frame filtering, while traffic shaping may be summarized by queuing frames, queue management, and transmission selection. Traffic policing is done by ingress filtering, egress filtering, as well as flow metering.

Q defines a queuing model at the transmission port where each port may implement up to eight traffic class queues. The queues have port-local priorities, with the lowest priority 0 to the highest priority 7. Traffic shaping in Q comprises how frames are assigned to queues and how and when frames are selected for transmission from said queues. In general, it is assumed that frames, once enqueued, do not change order within the said queue, i.e., the queues have first-in first-out service semantics. Pre-TSN (and pre-AVB) bridges basically used the priority code point of the VLAN tag of an Ethernet frame to determine the queue of a frame. Table 8–5 of the Q standard give a recommendation for the mapping of priority code point to traffic class queue and Annex I gives a rationale for this mapping as well as a discussion of actual use cases. Furthermore, in pre-TSN (and pre-AVB), the transmission selection from the queues mostly followed a strict priority selection scheme (i.e., queues with higher priority are served before queues with lower priority). TSN introduced additional means for both enqueueing and dequeuing to benefit real-time transmissions.

In switched Ethernet networks, there are two types of traffic-switching operation: store and forward, and cutthrough. Store and forward refers to the technique that an Ethernet frame is to be completely received at a reception port of a given bridge before the transmission of that Ethernet frame at any transmission port of the bridge may be initiated. Cutthrough, on the other hand, allows the start of transmission of an Ethernet frame even if the bridge did not yet completely receive the frame. Q is rather vague on whether cutthrough operation mode is a legitimate behavior of a bridge or not. As in industrial automation and motion control, there are many use cases for which cutthrough is essential to meet extremely low-latency requirements; there is an ongoing debate on whether to explicitly include cutthrough in TSN or to explicitly deny cutthrough and to formulate alternatives. In our following discussions of TSN, we will assume the store-and-forward mode of operation.

Table 1 Notation

Term	Short	Examples
device	D	end station / bridge
end station	ES	ES1, ES2
bridge	B	B1
link (directed)	[start,end]	[ES1, B1]
link (undirected)	(device, device)	(ES1, B1)
path	$[\mathrm{link1},..,\mathrm{link}_n]$	[[ES1,B1],[B1,ES2]]
frame (on link)	$\mathrm{f}_{\mathrm{name}}^{\mathrm{link}}$	$\mathrm{f}_1^{[\mathrm{ES1,\ B1}]}$
stream	$\mathrm{f}_{\mathrm{name}}$	f_1
port	$\mathrm{B.p}^{\mathrm{device}}$	$\mathrm{B1.p}^{\mathrm{ES1}}$
port out	$\mathrm{B.pout}_{\mathrm{queue}}^{\mathrm{end}}$	$\mathrm{B1.pout}_1^{\mathrm{ES2}}$
port in	$\mathrm{B.pin}^{\mathrm{start}}$	$\mathrm{B1.pin}^{\mathrm{ES1}}$
gate (Qbv)	$\mathrm{B.gbv}_{\mathrm{queue}}^{\mathrm{end}}$	$\mathrm{B1.gbv}_7^{\mathrm{ES2}}$
gate (Qci)	$\mathrm{B.gci}_{\mathrm{queue}}^{\mathrm{end}}$	$\mathrm{B1.gci}_7^{\mathrm{ES2}}$
frame schedule	<time,frame>	$<\mathrm{t}_{\mathrm{TT}},\mathrm{f}_{\mathrm{TT}}^{[\mathrm{ES1,\ B1}]}>$
gate schedule	<time,gate=(o,C)>	$<\mathrm{t}_{\mathrm{TT,B1.gbv}^7}^{\mathrm{ES2}}=\mathrm{o}>$

In the remainder of this paper, we will use the notation given in Table 1.

Whenever possible we will use as example a simple network of two end stations, ES1 and ES2, connected to each other via a single bridge, B1 (see Fig. 3). Following our notation, this network can be encoded as follows: ES1 connects to B1 on port B1.$\mathrm{p}^{\mathrm{ES1}}$ via the undirected link (ES1, B1), and ES2 connects to B1 on port B1.$\mathrm{p}^{\mathrm{ES2}}$ via the undirected link (B1, ES2). A transmission path from ES1 to ES2 may, therefore, be: [[ES1, B1], [B1, ES2]]. On this path, ES1 may send a frame $f_1^{[\mathrm{ES1,\ B1}]}$ to B1 and B1 may forward said frame as $f_1^{[\mathrm{B1,\ ES2}]}$ to the second end station ES2. For some mechanisms, we will use an extended network as an illustrative example.

There is a subtle difference between the concept of a frame and a stream[2]: a frame indicates a particular instance of a message on a particular link in the network, e.g., $f_i^{[\mathrm{ES1,\ B1}]}$ is frame i transmitted on the link [ES1, B1]. A stream is a concept of information distribution between a sender and a receiver (or multiple receivers); to keep matters simple, we will assume all sender and receiver devices being end stations. A stream groups all frames together that belong to such an information distribution. For example, in the simple network, ES1 may be the sender of a stream f_i and ES2 may be its receiver. The actual realization of the stream f_i is by at least the two frames $f_i^{[\mathrm{ES1,\ B1}]}$ and $f_i^{[\mathrm{B1,\ ES2}]}$. However, stream f_i may be a periodic information distribution and, therefore, cause periodically the transmissions of frames $f_i^{[\mathrm{ES1,\ B1}]}$ and $f_i^{[\mathrm{B1,\ ES2}]}$. Likewise, f_i may be aperiodic or sporadic also resulting in multiple frame transmissions on the network. All these frames are said to belong to the same stream. Typically, the stream identifier will be encoded in each frame. The IEEE 802.1CB standard defines various options on how to encode the stream identifier in a frame. A common approach in TSN is to encode the stream identifier in the Ethernet MAC destination address and the VLAN identifier. In TSN, the sender of a stream is referred to as a talker, while the receiver of a stream is referred to as a listener.

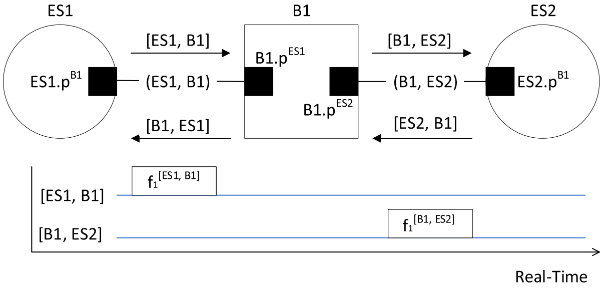


Fig. 3. ***Simple network.***

III. IEEE 802.1 TSN STANDARDS OVERVIEW

TSN is a collective name for a set of standards, standard amendments, and projects, published or under development by the TSN task group of the IEEE 802.1 Working Group. The TSN standards define mechanisms to provide deterministic services through IEEE 802 networks, such as guaranteed packet transport with bounded latency, low packet delay variation, and low packet loss. Fig. 4 lists the main TSN standards and projects (and other relevant standards).

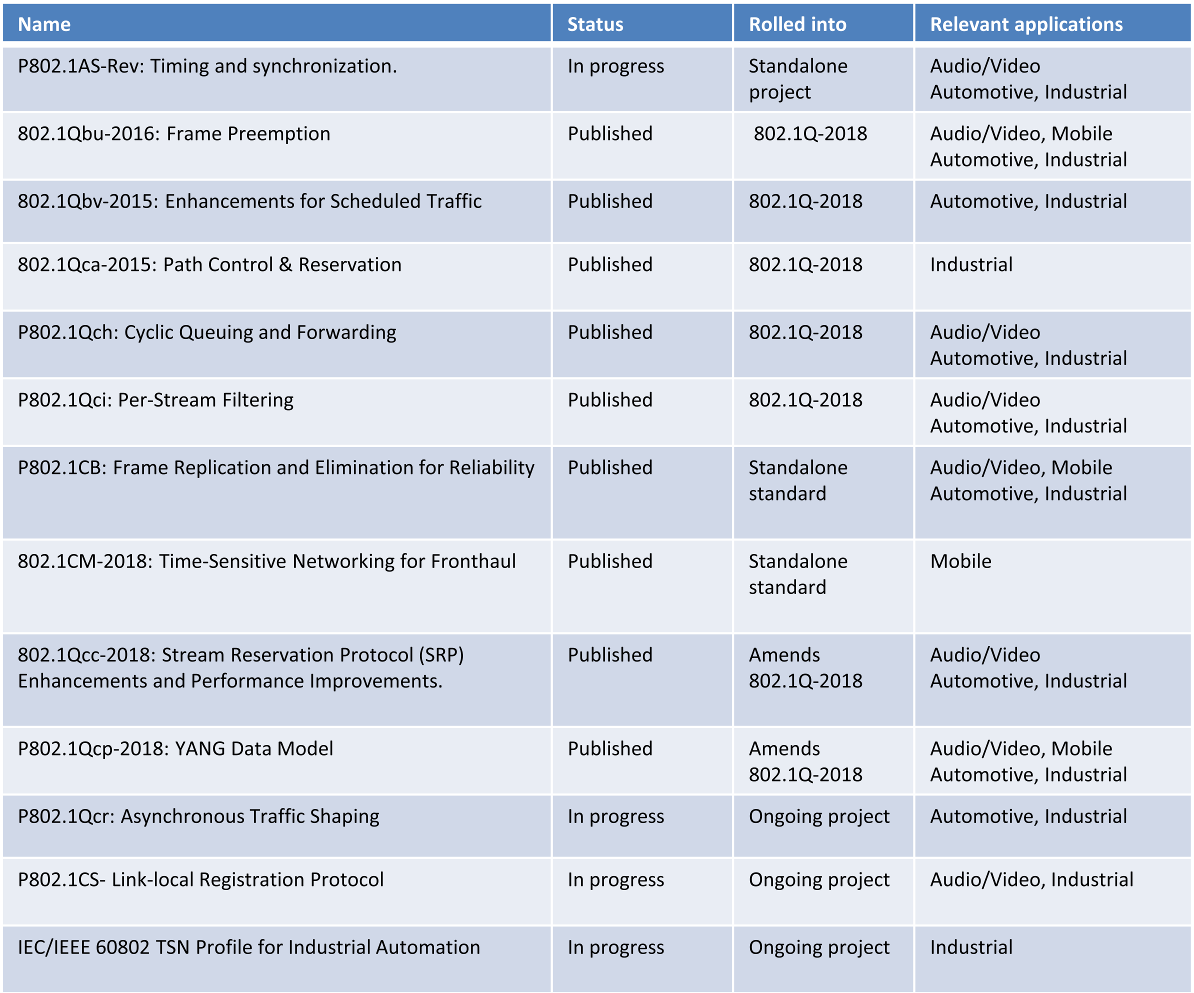
The TSN set of standards is a flexible toolbox (see Fig. 5 as an example depiction), from which a network designer can pick only what is needed for the targeted applications when it is needed. One way to survey the TSN standards is to arrange them in clusters based on the support they provide to the main pillars, i.e., timing and synchronization, bounded low latency, reliability, and resource management. However, such clusters are not disjoint, as some standards contribute to more than one aspect. For instance, the IEEE 802.1AS-Rev addresses time synchronization but it is also relevant to reliability.

A. Timing and Synchronization

Time synchronization plays a crucial role in all the applications targeted by 802.1 standards, albeit with different requirements when moving from home audio/video streaming to professional audio/video or time-sensitive and safety-critical control applications. For instance, highly and fully automated driving (AD) applications not only require a common notion of time for sensor fusion but also require redundancy to allow the system to remain operational and reach a minimal risk condition even in case of failure of a link or of the grandmaster [10]. This difference is reflected in the characteristics of the two relevant standards, the IEEE 802.1AS-2011 and the IEEE 802.1AS-Rev.

The IEEE 802.1AS-2011-Timing and Synchronization for Time-Sensitive Applications standard is one of the three 802.1 AVB standards and it targets networked audio/video applications. The standard specifies a level-2 profile of the

[2] We use "stream" and "flow" synonymously in this paper.



Name	Status	Rolled into	Relevant applications
P802.1AS-Rev: Timing and synchronization.	In progress	Standalone project	Audio/Video Automotive, Industrial
802.1Qbu-2016: Frame Preemption	Published	802.1Q-2018	Audio/Video, Mobile Automotive, Industrial
802.1Qbv-2015: Enhancements for Scheduled Traffic	Published	802.1Q-2018	Automotive, Industrial
802.1Qca-2015: Path Control & Reservation	Published	802.1Q-2018	Industrial
P802.1Qch: Cyclic Queuing and Forwarding	Published	802.1Q-2018	Audio/Video Automotive, Industrial
P802.1Qci: Per-Stream Filtering	Published	802.1Q-2018	Audio/Video Automotive, Industrial
P802.1CB: Frame Replication and Elimination for Reliability	Published	Standalone standard	Audio/Video, Mobile Automotive, Industrial
802.1CM-2018: Time-Sensitive Networking for Fronthaul	Published	Standalone standard	Mobile
802.1Qcc-2018: Stream Reservation Protocol (SRP) Enhancements and Performance Improvements.	Published	Amends 802.1Q-2018	Audio/Video Automotive, Industrial
P802.1Qcp-2018: YANG Data Model	Published	Amends 802.1Q-2018	Audio/Video, Mobile Automotive, Industrial
P802.1Qcr: Asynchronous Traffic Shaping	In progress	Ongoing project	Automotive, Industrial
P802.1CS- Link-local Registration Protocol	In progress	Ongoing project	Audio/Video, Industrial
IEC/IEEE 60802 TSN Profile for Industrial Automation	In progress	Ongoing project	Industrial

Fig. 4. ***Overview of the TSN standards.***

IEEE 1588 Precision Time Protocol (PTP) which includes the transport of synchronized time over bridged and virtual bridged local area networks, the selection of the timing source, and the notification of timing impairments, such as phase and frequency discontinuities. It also defines synchronization over other media (e.g., IEEE 802.11). The IEEE 802.1AS-2011 standard provides precise time synchronization of the network nodes. Implementations of IEEE 802.1AS-2011 have demonstrated to achieve a reference time with an accuracy better than 1 μs. Although the loss of the active grandmaster is handled by electing a new one, the handover procedure in this standard is not instantaneous and might cause time jumps at some nodes.

To cope with this issue, the IEEE 802.1AS-Rev targets the maintenance of synchronized time for time-sensitive applications (such as professional audio/video and time-sensitive control) across IEEE 802 networks during normal operation, and following addition, removal, or failure of network components and network reconfiguration. In the standard, redundancy is improved allowing for configuring multiple grandmaster clocks and multiple synchronization spanning trees to enable seamless low-latency handover and quick recovery of time synchronization in failure modes.

Both these standards are extensively discussed in Section IV-A.

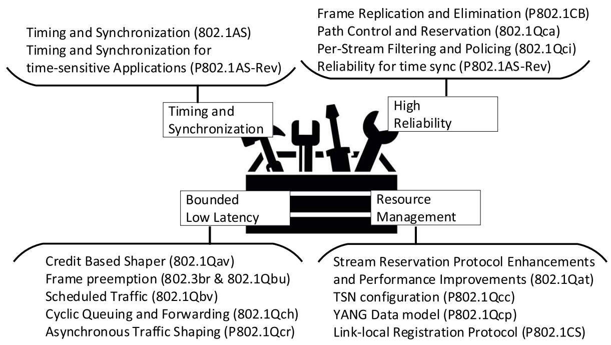

Timing and Synchronization (802.1AS)
Timing and Synchronization for
time-sensitive Applications (P802.1AS-Rev)
Frame Replication and Elimination (P802.1CB)
Path Control and Reservation (802.1Qca)
Per-Stream Filtering and Policing (802.1Qci)
Reliability for time sync (P802.1AS-Rev)
Timing and Synchronization
High Reliability
Bounded Low Latency
Resource Management
Credit Based Shaper (802.1Qav)
Frame preemption (802.3br & 802.1Qbu)
Scheduled Traffic (802.1Qbv)
Cyclic Queuing and Forwarding (802.1Qch)
Asynchronous Traffic Shaping (P802.1Qcr)
Stream Reservation Protocol Enhancements
and Performance Improvements (802.1Qat)
TSN configuration (P802.1Qcc)
YANG Data model (P802.1Qcp)
Link-local Registration Protocol (P802.1CS)

Fig. 5. ***IEEE 802.1 TSN toolbox.***

B. Bounded Low Latency

Guaranteed delivery of messages with real-time constraints is one of the main improvements of AVB and TSN to standard IEEE 802.1 networks including Ethernet. The IEEE 802.1-Qav Forwarding and Queuing Enhancements for time-sensitive streams standard has already been developed as part of the AVB suite of performance improvements and defined protocols and mechanisms that guarantee real-time transmissions even without global synchronization of the end stations and bridges. TSN even further improves the real-time performance by two published standards: IEEE 802.1Qbv and IEEE 802.1Qbu. IEEE 802.1Qbv defines schedule-driven communication, i.e., to leverage synchronized time in transmission and forwarding decisions for messages in the network. IEEE 802.1Qbu standardizes a preemption mechanism that allows time-critical messages to interrupt ongoing nontime-critical transmissions. The IEEE 802.1Qbv standard is addressed in detail in Section IV-B, whereas the IEEE 802.1Qbu is addressed in detail in Section IV-C.

For completeness, we highlight another standardization project currently being executed to standardize nonsynchronized real-time communication: IEEE 802.1Qcr standardizes asynchronous traffic shaping based on the concepts introduced in [11].

C. Reliability

One of the main requirements that make the difference between AVB applications and TSN applications is the need for highly reliable packet delivery. For example, autonomous driving systems include delay-sensitive real-time applications (e.g., machine vision) that cannot tolerate the delay due to retransmissions of lost frames. These applications also demand prompt detection of error conditions, such as traffic overload, and measures to prevent error propagation over the network. Several TSN standards contribute to communication reliability, addressing it from different perspectives, such as packet duplication and elimination to reduce the packet loss probability, path control and bandwidth reservation, detection of misbehaving streams and enforcement of mitigating actions, and configuration. These standards are summarized in the following.

The IEEE 802.1CB—Frame Replication and Elimination for Reliability standard defines protocols for bridges and end stations that provide redundant packet transmission over separate paths through the network and elimination of duplicate packets at or nearby the destination, so as to decrease the packet loss probability. The protocol also provides sequence numbering of packets and multiple stream identification functions. The IEEE 802.1CB is addressed in detail in Section IV-E.

The IEEE 802.1Qca—Path Control and Reservation standard is an extension of intermediate system-to-intermediate systems (IS-IS) shortest path bridging that offers explicit forwarding path control, thus enabling the use of the nonshortest paths. It also integrates a tool for bandwidth and stream reservation along the forwarding path and resiliency control mechanisms for data traffic. Since the first realizations of TSN in industrial settings will be engineered networks, i.e., their logical topology will be statically defined at network design time, we do not focus on the IEEE 802.1Qca in this paper.

The IEEE 802.1Qci—Per-Stream Filtering and Policing (PSFP) standard defines procedures to make filtering decision and enforce policing on a per-stream basis. It offers quality of service protection to streams, allowing to detect if some streams are not conforming to the behavior agreed by configuration and/or protocol exchanges and taking mitigating actions. The IEEE 802.1Qci standard is addressed in detail in Section IV-D.

D. Resource Management

Resource reservation and management are key aspects to achieve deterministic networking. Multiple TSN standards contribute to bandwidth reservation and management and they accomplish this in different ways. For instance, to fulfill requirements such as predictable behavior and zero packet loss, it is required to establish and enforce a bandwidth contract between the network and the application. This is realized by limiting the source of a TSN flow to comply with a maximum packet size and the maximum number of packets transmitted per time intervals. In this way, all needed network resources can be reserved for specific traffic streams traversing a bridged local area network. This is the task of the stream reservation protocol (SRP) in the IEEE 802.1Qat standard and of its enhancements in the IEEE 802.1Qcc standard.

The IEEE 802.1Qat—SRP (rolled into IEEE 802.1Q-2014) is one of the AVB standards. It allows for the registration and reservation of resources, such as buffers and queues, within bridges along the path between the talker and the listener, so as to enable the E2E management of quality of service for streams with latency and bandwidth guarantees.

The IEEE 802.1Qcc—SRP Enhancements and Performance Improvements standard defines protocols to support configurable stream reservation classes and streams. It amends the IEEE 802.1Qat standard extending the capabilities of SRP so as to fulfill the requirements of professional audio/video, industrial, consumer, and automotive markets. The IEEE 802.1Qcc standard is addressed in detail in Section IV-F.

Another standard that is relevant to resource management is the IEEE 802.1CS—Link-local Registration Protocol, which defines procedures to replicate a large registration database (in the order of 1 MB) from one end to the other of a point-to-point link. IEEE 802.1CS aims to facilitate the creation of application protocols that distribute information through all or part of a network. IEEE 802.1CS arises from the needs of new applications, such as industrial automation or audio/video for large venues, which require much more configuration/registration/reservation data than those that the current 802.1Q Multiple Registration Protocol (MRP) can support. In fact, MRP is optimized for databases up to 1500 bytes and slows significantly when used for larger databases.

IV. CORE STANDARDS FOR CRITICAL COMMUNICATION

A. Clock Synchronization IEEE 802.1AS/.1ASrev

The synchronization of local clocks in a distributed system has many well-understood benefits. For example, the transmission of frames in a network may be coordinated to reduce transmission latency and jitter (see the next section on IEEE 802.1Qbv); applications may use a synchronized timebase for a timestamping mechanism of critical events. End stations may also use synchrony to coordinate the execution of tasks, e.g., to cause a synchronized reading of sensor values throughout the distributed system. Thus, TSN defines a suite of clock synchronization protocols in IEEE 802.1AS (AS for short) such that end stations and bridges may synchronize their local clocks to each other. These protocols are often also referred to as generalized PTP (gPTP). AS is discussed in detail by Garner and Ryu [12] and Stanton [13]. We will give an overview of AS as well as some discussion of key items of the AS revision ASrev next.

AS standardizes solutions to the following problems: determination of a synchronization hierarchy in the network, distribution of the time from the one or many root nodes in the hierarchy to the rest of the network, and the measurement of link delays between devices. AS is currently under revision by the TSN task group and we will refer to this revised version by ASrev. In the following, we use the simple network of two end stations connected to each other via a single bridge in the discussion of illustrative examples of the AS clock synchronization protocols.

1) Protocol Overview: AS specifies a leader–follower paradigm with leader election: a master, called the grandmaster, acts as the reference time for the other devices in the network and the best master algorithm automatically selects the one grandmaster device from a potentially larger set of grandmaster-capable devices. Then, the algorithm dynamically configures the synchronization hierarchy, called the synchronization spanning tree, in the network. The synchronization spanning tree is a minimum spanning tree in the network and is realized by assigning to each port in the network one of three states: master, slave, or passive. Ports may also be disabled entirely by setting the port synchronization state to a disabled state. Therefore, for example, in the simple network (see Fig. 6), ES1 may be elected the grandmaster clock. Then, ES1.p^{B1} (the port connecting ES1 to B1) is set to master state, B1.p^{ES1} is set to slave state, B1.p^{ES2} is set to master state, and ES2.p^{B1} (the port connecting ES2 to B1) is set to slave state. Any other ports in the devices, e.g., in case B1 would provide further ports, would be set to the passive state.

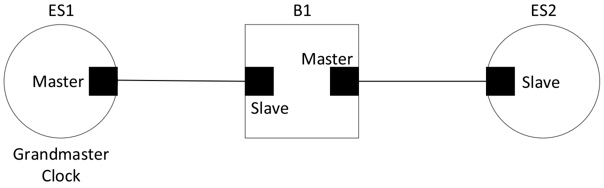


Fig. 6. *Simple network—as example.*

ASrev extends the capabilities of the synchronization hierarchy in the following ways: grandmasters and synchronization trees can be configured redundantly to achieve increased fault-tolerance and the synchronization trees may be configured explicitly without the invocation of the best master algorithm. Redundancy is enabled by the ability to configure multiple gPTP domains. Conceptually, each gPTP domain is a separate instantiation of the gPTP protocols, i.e., the devices in the network execute multiple instances of gPTP. However, the standards do give some guidance to leverage synergies for a cost-efficient realization of multiple gPTP domains, e.g., link-delay measurement can be done once for multiple gPTP domains.

Once the synchronization hierarchy is established, AS defines how to distribute the time from the root (i.e., the grandmaster) to the other devices in the network. The protocol is executed periodically and starts with the grandmaster transmitting a synchronization message f_{SYNC} on all ports with port synchronization state set to master. When a device receives f_{SYNC} (as per definition on a port with synchronization state set to slave state), it takes a timestamp of the reception point in time and initiates transmission of new f_{SYNC} frames on all ports set in the master state. Again, the device takes a timestamp upon transmission f_{SYNC} from a port. By means of the timestamps, each device is aware of the so-called residence time of f_{SYNC} within the device (or multiple residence times, in case the device had multiple ports in the master state), by simply subtracting the reception timestamp from the transmission timestamp. As f_{SYNC} propagates further in the synchronization hierarchy, i.e., along the synchronization spanning tree, each further device in the tree executes this process, timestamping and forwarding f_{SYNC}, until devices that do not have ports in master state receive f_{SYNC}, i.e., the leaf nodes in the synchronization spanning tree (ES2 in the simple network example).

At this point of the protocol, all devices in the network perceived a mutual resynchronization event (the reception of f_{SYNC}) which already synchronizes the local clocks of the devices to each other. For example, devices executing a simplified version of the AS protocol could simply reset their local clocks to 0 whenever they receive f_{SYNC}. However, this simplified protocol would have two main drawbacks: first, the time horizon would be bound by the frequency of the f_{SYNC} distribution, i.e., the local clocks could measure only relatively short time durations before they were reset to 0. Second, this synchronization according to the simplified protocol may be of poor quality, especially in larger networks, as it does not take the transmission latencies of f_{SYNC} into account. Since devices close to the grandmaster receive f_{SYNC} earlier than devices positioned farther away, the closer devices reset their clocks earlier than the other devices.

AS solves both issues by the exchange of a second message, the followup frame f_{FLUP}. f_{FLUP} is, again, initiated by the grandmaster and distributed along the synchronization spanning tree throughout the network. To counter the first issue described earlier, f_{FLUP} carries the preciseOriginTimestamp, the point in time when the grandmaster initiated the transmission of f_{SYNC}. This point in time would typically be with respect to a much longer time horizon than the period of the f_{SYNC} transmissions; there will be use cases in which time is synchronized to UTC, in which case time will never wrap around during system operation. The second issue of the simplified protocol is mitigated by f_{FLUP} accumulating the residence times in a so-called correctionField of the frame. Thus, as f_{FLUP} traverses the synchronization spanning tree, the correctionField gets updated from device to device with the individual times f_{SYNC} resided in the respective device.

The collection and distribution of the residence times already improve the synchronization quality of AS over the simplified version significantly, but AS even implements further features to fine-tune the synchronization quality. First, another protocol continually measures the latencies on the communication links at runtime (as opposed to setting static parameters, like cable lengths). Second, a procedure is in place to directly measure relative clock drifts of neighboring devices and, based on said measurement, to indirectly measure the relative clock drift of the devices to the grandmaster clock. The measured link latencies are also added to the correctionField of f_{FLUP}, while the measured relative clock drift is used to normalize the time value in the received f_{FLUP} and a device's residence time before addition.

2) Protocol Discussion and Open Issues: A key performance indicator of a clock synchronization protocol is the quality to which the distributed local clocks in the system can be synchronized to each other. Especially in large-scale industrial production lines, clock synchronization protocols need to be carefully engineered. In a recent simulation study, Gutiérrez *et al.* [14] evaluated the impact of an exhaustive list of physical parameters on the synchronization quality of AS. The simulation study concluded that AS manages to synchronize local clocks to each other with a maximum synchronization difference of 1 μm in topologies with up to 30 hops and a maximum synchronization difference of 2 μm in topologies with up to 100 hops.

Another performance indicator of clock synchronization protocols is their reaction to security attacks. AS (and its related standard IEEE 1588) has been subject to various security studies (IEEE 1588 actually provides a security annex). Examples of security analyses in this area are [15] and [16]. The Internet Engineering Task Force (IETF) [17] even standardizes a threat model and security requirements for clock synchronization protocols.

While many threats can be mitigated by typical cybersecurity procedures such as encryption or cryptographic signatures, a recent study by Lisova *et al.* [18] addresses delay attacks that are difficult to mitigate by traditional measures. In such a delay attack, a bridge in the synchronization spanning tree is infiltrated by an attacker and the forwarding process of f_{SYNC} and/or f_{FLUP} are compromised. In particular, said bridge may lie about the residence time of f_{SYNC} and add too long or too short values to the correction field f_{FLUP}. Consequently, devices that receive this compromised f_{FLUP} (and all devices that follow in the synchronization spanning tree) will suffer from poor synchronization quality. Even worse, these devices may not even be aware that their local clock readings are significantly off from the grandmaster clock. Lisova *et al.* [18] continue to propose intrusion detection systems for this kind of faults as a mitigation strategy.

Finally, clock synchronization algorithms are also assessed with respect to their ability to generally tolerate failures of devices and connections. Such failures may be a consequence of hardware faults or software bugs and different industries have different views on which failure semantics justifiably can be assumed for a failing device. For example, in the aerospace domain, typically a Byzantine failure semantics is assumed for chips that exceed a certain level of complexity (e.g., chips with more than a couple of hundred logic gates). Thus, we would have to assume that in the failure case, a bridge may generate arbitrary frames at arbitrary times with arbitrary frequencies. Other industries allow more benign failure semantics. Fault-tolerant clock synchronization is a research discipline since at least the 1980s with enlightening papers [19]. AS/ASrev standardizes basic infrastructure mechanisms, such as to distribute time from multiple grandmasters via redundant clock synchronization trees, but does not go further to define how to construct a network such that particular failure assumptions can be tolerated. Furthermore, AS/ASrev does not address the initial synchronization of the local clocks (the startup phase) nor a system-wide restart phase. However, it seems plausible that existing truly fault-tolerant clock synchronization protocols like SAE AS6802 [20] could be implemented on top of an ASrev system.

B. Schedule-Driven Communication IEEE 802.1Qbv

One use case of synchronized local clocks in a distributed system is schedule-driven communication, often referred to as time-triggered (TT) communication. The core principle of TT communication is rather simple: the system designer generates a communication schedule that instructs the end stations when to send which frames to the network, this communication schedule (or parts of it) is distributed to the end stations and bridges as part of their configuration, and a scheduling function at the end station (and also in bridges in case of TSN as we will elaborate in this section) executes the communication schedule. For example, in the simple network, the communication schedule may be of the form $\langle t_{\mathrm{TT}},\ f_{\mathrm{TT}}^{[\mathrm{ES1,\,B1}]}\rangle$ and instruct ES1 to send a frame $f_{\mathrm{TT}}^{[\mathrm{ES1,\,B1}]}$ at a point in time t_{TT} to B1.

TT communication comes in many flavors (e.g., Profinet [21], TT Ethernet [22], FTT-Ethernet [23], TTP [24], and FlexRay [25]). With IEEE 802.1Qbv (Qbv for short), TSN has also introduced a variant of TT communication to the IEEE 802.1 set of standards, which we discuss in this section.

1) Mechanism Overview: As a key concept unique to IEEE 802.1, Qbv does not directly schedule frame transmissions (as described in the generic example mentioned above) but schedules the activation and deactivation of queues. Said queues are both the traffic class queues in bridges as introduced earlier (see Section II) as well as the transmit queues in end stations. Qbv introduces a gate per queue as a means to realize the activation and deactivation of a queue. The gate is in the open state to activate the queue; alternatively, the said gate is in the closed state to deactivate the queue. The transmission selection process at the port of a bridge may only select frames from queues that are activated (i.e., have their gates in the open state). Although Qbv requires the same mechanism in the end station, for simplicity, we may assume that in the end station, indeed, the frame transmissions are scheduled (which will also be the likely case for end station implementations). This is generally deemed acceptable since the internal operation of an end station does not need to be disclosed and at the end station's interface to the network frame transmission scheduling cannot be distinguished from queue-based scheduling in the first place.

The communication schedule in Qbv is realized by a gate control list (GCL) where the entries in the list indicated for each port and traffic class the points in time when to set the gate state into the open/closed state.[3] In the simple network, an entry in such a GCL for bridge B1 may look like the following: $\langle t_{\mathrm{TT}}, \mathrm{B1.gbv}_7^{\mathrm{ES2}} = o\rangle$. It instructs B1 to set the queue 7 on the port that connects B1 to ES2 in the open state at the point in time t_{TT}. Consequently, the transmission select process at this port may select frames assigned to said queue from t_{TT} onward. In case a succeeding entry in the GCL sets the gate state into the closed state, the transmission select must stop to select messages from this queue a sufficiently long duration before this gate closing event, such that it is guaranteed that a selected message can be completely transmitted before the gate closes. Thus, it is guaranteed that when the gate of second queue transitions from the closed to the open state at the very same instant when said first gate closes, messages from the closing queue do not delay messages from the opening queue. This critical time before a gate closing event is referred to as the guardband. Because of this freedom of interference of queues on each other, sometimes the time windows that are generated by matching opening and closing events of the gate of a queue are referred to as protected transmission windows. To achieve this effect, a simple implementation of a bridge may stop transmitting messages from the said queue as long as the duration of a maximum-sized message before the gate closes. A more sophisticated implementation may decide dynamically whether messages in the queue fit for transmission or not (as time progresses toward the gate closing event the permittable frames become shorter and shorter). Qbv also defines a specific mechanism (HOLD and RELEASE) when Qbv is to be used together with frame preemption, which we discuss in the following section.

To illustrate the effect of Qbv, let us extend the simple network by another end station, ES3, which connects to B1 as well (see Fig. 7). Furthermore, let us assume that both ES1 and ES3 need to communicate with ES2, which acts purely as a sink for information in this example. ES1 is synchronized to B1 and the information of ES1 is time-critical (the information of ES3 is not) and must never be delayed by frames originating from ES3. Finally, we assume that

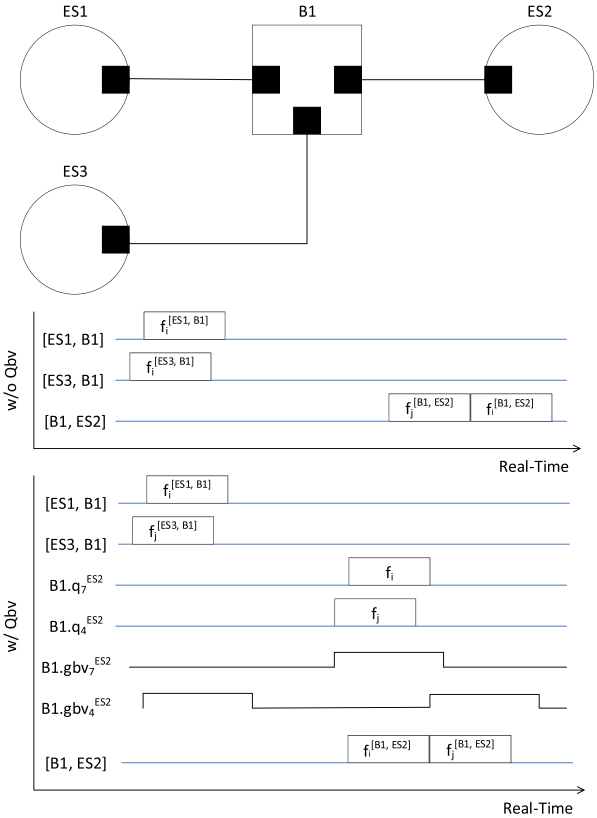


Fig. 7. ***Extended network—Qbv example.***

[3] In fact, Qbv standardizes the relative duration in between gate state changes in the GCL. Since this is functionally equivalent to absolute time stamps and the description of absolute time stamps is more intuitive, we have chosen the later for the following discussion.

ES3 is not capable to synchronize (e.g., by means of AS) and, therefore, may initiate transmissions at any point in time. The traditional approach of IEEE 802.1 to resolve this problem would be to assign frames $f_j^{[\text{ES3, B1}]}$ from ES3 lower priority code points than frames $f_i^{[\text{ES1, B1}]}$. As a result, $f_j^{[\text{ES3, B1}]}$ would be enqueued in a lower priority traffic class queue than $f_i^{[\text{ES1, B1}]}$, which would get priority by transmission select in case both frames are available for selection at the same point in time, for example, $f_j^{[\text{ES3, B1}]}$ goes to B1.gbv$_4^{\text{ES2}}$ and $f_i^{[\text{ES1, B1}]}$ is enqueued in B1.gbv$_7^{\text{ES2}}$. However, in case that $f_j^{[\text{ES3, B1}]}$ would be ready for transmission select slightly earlier than $f_i^{[\text{ES1, B1}]}$, transmission select would start to transmit f_j as $f_j^{[\text{B1, ES2}]}$ and thereby delay f_i (which will be transmitted as $f_i^{[\text{B1, ES2}]}$ later). Unfortunately, since ES3 operates unsynchronized (as per our assumption), the described scenario is rather likely to occur and using only non-TSN mechanisms we are unable to achieve our originally stated goal, i.e., that ES3 transmissions shall never delay ES1 transmissions.

Qbv solves this issue. As ES1 is synchronized to B1, we can assign a high traffic class (e.g., 7) to $f_i^{[\text{ES1, B1}]}$ and a low traffic class (e.g., 4) to $f_j^{[\text{ES3, B1}]}$ and construct a GCL that sets the gate of the low traffic class to the closed state a sufficiently long duration before $f_i^{[\text{ES1, B1}]}$ becomes ready for transmission select in B1. That means there must be a gate schedule entry $\langle t_{\text{TTx}}, \text{B1.gbv}_4^{\text{ES2}} = C \rangle$, that brings the port state of traffic class 4 into the closed state at an appropriate time t_{TTx}. The gate of the high traffic class may always be set in the open state in this example but must be set into open sufficiently long before $f_i^{[\text{ES1, B1}]}$ becomes ready for transmission select in B1.

2) Mechanism Discussion and Open Issues: While the Qbv mechanism in itself is rather simple, there is an inherent complexity in generating the GCLs, i.e., finding the right points in time when to open and when to close the gates. Indeed, in general, the synthesis of communication schedules in real-world scenarios turns out to be often NP-complete: there is no known algorithm that generates the communication schedule in polynomial time. However, the schedule synthesis is an active field of research and many results are transferable to Qbv. We can differentiate two main types of research methodology in the synthesis of communication schedules: a first type aims to construct specialized search algorithms, e.g., by deploying heuristics, metaheuristics, or genetic algorithms and a second type leverages general purpose tools, such as integer linear programming (ILP) or satisfiability modulo theories (SMT) solvers.

Examples of the first type are: Tămaş-Selicean *et al.* [26], Hanzálek *et al.* [27], and Tanasa *et al.* [28]. Examples of the second type are: Zeng *et al.* [29] and Steiner [30].

As Qbv is a rather novel standard, there are only a few contributions that directly target Qbv scheduling. Craciunas *et al.* [31] demonstrate how to translate scheduling approaches from TTEthernet to TSN. Craciunas *et al.* [32] and Serna Oliver *et al.* [33] propose a formal set of scheduling constraints that directly schedule the GCLs of Qbv and report the first schedule synthesis performance numbers in [34] based on SMT.

C. Frame Preemption IEEE 802.1Qbu and IEEE 802.3br

1) Protocol Overview: The IEEE 802.1Qbu—Frame Preemption standard, now enrolled into the IEEE 802.1Q-2018 standard, defines procedures for bridges and end stations to hold or suspend the transmission of a frame in order to allow the transmission of one or multiple more urgent frames, as well as procedures to release or resume the transmission of the less critical frame once the urgent frames have been transmitted.

The Qbu standard is connected to the IEEE 802.3br—Specification and Management Parameters for Interspersing Express Traffic standard, which allows urgent time-critical data frames to split into smaller fragments the noncritical frames in transit over a single physical link. To this aim, the IEEE 802.3br provides two MAC service interfaces as follows.

1) Preemptable media access control (pMAC) for preemptable frames.
2) Express media access control (eMAC) for preemptive frames that are called express frames.

The IEEE 802.3br standard introduces an optional sublayer, called a MAC Merge sublayer, which attaches an eMAC and a pMAC to a single PHY through a reconciliation sublayer. The PHY is unaware of preemption. The MAC merge sublayer and its MACs provide support for frame preemption as defined in IEEE Std 802.1Qbu. In particular, the MAC merge sublayer provides two different ways to hold transmission of preemptable traffic in the presence of express traffic. One is interrupting (preempting) the preemptable traffic being currently transmitted, and the other one is preventing preemptable traffic from starting transmission.

As far as preemption is concerned, it is worth noticing that the ongoing transmission of a preemptable frame can be interrupted only if: 1) at least 60 octets of the frame have been transmitted and 2) there are at least 64 octets left to be transmitted [including the frame cyclic redundancy check (CRC)]. More specifically, 64 octets are the minimum final fragment size. A receiver can request that any nonfinal fragments are larger than this by 0, 64, 128, or 192 octets. However, the typical value for the minimum nonfinal fragment size is 64 octets. For error correction purposes, each nonfinal fragment carries a particular CRC value (mCRC value) that is used to determine the correct reception of all the frame octets up to the last octet transmitted in that fragment, whereas the CRC field of the final fragment contains the last 4 octets of the preempted MAC frame [i.e., the frame check sequence (FCS) field]. Frame count and fragment continuation mechanisms are provided to detect packet loss and to avoid reassembly of invalid fragments.

As far as the second way is concerned, i.e., preventing preemptable traffic from starting transmission, the IEEE 802.3br standard defines a MAC merge service interface (MMSI) and a primitive (MM_CTL.request) that causes the transmission of preemptable traffic to be held or released. The single parameter of the primitive (hold_req) takes one of the two values, HOLD or RELEASE. In the first case, the MAC merge sublayer holds back preemptable traffic, preventing it from beginning transmission. In the second case, it allows the transmission of preemptable traffic when the eMAC does not have a packet to transmit. Express traffic has a higher priority than preemptable traffic. Regardless of whether the preemption capability is active or not, the MAC merge sublayer allows express traffic or the MMSI service primitive to prevent the transmission of preemptable traffic. The hold-and-release feature allows to minimize the latency experienced by express traffic. With disabled preemption, if both the eMAC and pMAC have a frame ready to transmit and no frame is being transmitted, the eMAC frame is chosen. However, if a pMAC frame is being transmitted and the eMAC has a frame ready to transmit, the latter is transmitted only after the completion of the ongoing pMAC transmission. An Ethernet with IEEE 802.3br capabilities can be exploited to support the fronthaul and backhaul of radio traffic over the Ethernet [35].

The preemption capability is enabled in transmission only after it has been verified that the other end of the link also supports preemption. The assessment is made during a verification phase, which follows a negotiation based on the link layer discovery protocol [36] and the exchange of the additional Ethernet capabilities type length value.

IEEE 802.1Qbu amends the IEEE 802.1Q standard introducing support for frame preemption. In particular, each traffic class queue supported by each port of a bridge or end station is assigned one frame preemption status value between the express or preemptable via a frame preemption status table. Preemptable frames shall be transmitted using the pMAC service, while express frames through the eMAC service. All the priorities that map to the same traffic class have the same preemption status value. There is one frame preemption parameter table per port of a bridge or end station. The preemption values can be changed by management. The default value of frame preemption status is expressed for all the priority values.

According to Qbu, in a port of a bridge or station that supports frame preemption, a frame of a given priority is not eligible for transmission if the frame preemption status for that priority is preemptable and either a request to hold back the preemptable frame (holdRequest) has been issued to the MAC or the transmission of a previous preemptable frame has not completed yet due to preemption by an express frame.

Any of the available transmission selection algorithms in 802.1Q-2014, such as the credit-based shaper can be used with preemption enabled. The Qbu standard specifies the behavior of the credit-based shaper algorithm when used in combination with frame preemption. In particular, the shaper behaves in such a way that any delay introduced by preemption does not consume credits (e.g., if a frame is preemptable and its transmission is prevented or interrupted by an express frame belonging to a different queue, the transmit parameter of the shaper takes the FALSE value until the frame transmission begins or is resumed). In addition, no frame overhead associated with preemption is subtracted from the bandwidth reserved for the shaper. The Qbu standard also defines gate operation mechanisms for interrupting/resuming the transmission of preemptable frames when the enhancements for scheduled traffic provided by the Qbv are also enabled.

2) Protocol Discussion and Open Issues: Preemption is beneficial because the critical data do not have to wait until the full frame of noncritical data has been transmitted, thus reducing the delay experienced by time-critical frames due to the interference of noncritical traffic. However, the advantage of preemption depends on the operating speed, being greater at lower speeds. In fact, the upper bound on the medium access delay for an express frame in case of no preemption is the transmission time of a maximum-length noncritical frame. While such a time duration may be considerable at lower speeds, it decreases at higher operating speeds, so the benefit derived from preemption also decreases. Under high speeds, the work in [37] assessed, through NS3 simulations, the capability of Ethernet with preemption to support the stringent jitter requirements of Common Public Radio Interface (CPRI). The results proved that preemption can reduce the jitter but not enough to meet the jitter requirement. Conversely, the simulation results showed that IEEE 802.1Qbv, when configured with the local scheduling algorithm that is proposed in the paper, can cope with the jitter requirement for all the flows, although at the expenses of an increased delay for all of them. This result is interesting for CPRI; however, the proof-of-concept implementation used in this paper is not fully compliant with the 802.1Qbv standard.

IEEE 802.3br does not enforce a limit on how long a preemptable frame can be preempted by express traffic. A formal timing analysis, based on the compositional performance analysis (CPA), for frame preemption according to the IEEE 802.3br is presented in [38]. The work evaluates the effect of preemption on the worst case E2E latency guarantees of both standard Ethernet (IEEE 802.1Q) and TSN (IEEE 802.1Qbv). Equations to compute the maximum frame latency for preemptable flows are provided in the work, which also presents a simulation study based on realistic automotive traffic classes. The obtained results prove that preemption is very beneficial to control traffic (i.e., time-critical traffic) even on the standard Ethernet, which achieves worst case latency performance comparable with those of Ethernet TSN. This result, in the authors' view, paves the way for using standard Ethernet with preemption as an alternative to IEEE 802.1Qbv. However, the simulation results in [38] refer to the specific simulation scenarios considered in that work, depending on the type of flows and on their mapping on the traffic classes (preemptable/express) and priorities, so it is not necessarily of general validity. Preemption is a solution to real-time communication as long as the volume of express traffic is below an acceptable threshold: express traffic can be delayed by other express traffic. As long as this delay is acceptable for an application, preemption suffices as real-time communication solution for TSN. However, when the volume of express traffic exceeds said threshold, then the delay of express traffic on each other causes at least some of the express traffic to violate its latency requirements. Thus, in such cases, preemption alone is not sufficient anymore to guarantee real-time communication and other TSN mechanisms, like Qbv, are required.

According to the IEEE, the maximum bridge transit delay, i.e., the maximum time between the reception and the transmission of the same frame by the bridge, is equal to 1 s. To comply with this requirement, when preemption is combined with scheduling, a possible approach is configuring the express traffic windows so that their duration is much smaller than 1 s and the gaps between them are large enough to accommodate the transmission of a maximum size frame. This setting is feasible, as urgent frames typically are time-sensitive frames with a smaller size than less critical ones. When preemption is applied without scheduling, there are no express traffic windows. In this case, the limiting effect on express traffic of a traffic shaper, such as the credit-based shaper, can help to reduce the transit delay for preemptable frames.

Frame preemption can be used together with schedule-driven communication as defined in Qbv, and one common configuration use case will be to assign preemptable messages to a queue with a gate that is always in the open state. Thus, an express frame may only be delayed by the maximum-sized fragment of a frame (i.e., 124 octets). However, for some applications, even this delay may be unacceptable and, thus, Qbv defines a specific mechanism to combine schedule-driven communication with frame preemption even more efficiently: the Set-And-Hold-MAC and Set-And-Release-MAC gate operations.

These gate operations are also scheduled with respect to the synchronized time (such as gate opening and gate closing events) but have effect only on preemptable messages: Set-And-Hold-MAC inhibits the transmission of preemptable frames, while Set-And-Release-MAC continues the transmission of preemptable frames. Thus, scheduling a Set-And-Hold-MAC gate operation 124 octets before the gate open event of a time-critical traffic class ensures that no fragment of a preemptable frame will delay messages from the time-critical traffic class.

D. Traffic Filtering and Policing IEEE 802.1Qci

1) Mechanism Overview: The IEEE 802.1Qci—PSFP (Qci) standard, now enrolled into the IEEE 802.1Q-2018 standard, defines protocols and procedures to make filtering, policing, and queuing decisions on a per-stream basis, exploiting the stream identification functions specified in the IEEE Std 802.1CB standard. Qci implements the following three tables that can be modified by the network management functions.

1) Stream filter instance table.
2) Stream gate instance table.
3) Flowmeter instance table.

The stream filter instance table, which is an ordered list of stream filters, defines the filtering and policing actions to be applied to the frames of a specific stream. Stream filters make the stream go through a stream gate to a given flowmeter and then to an output queue.

Each stream filter contains several per-stream parameters, such as the following.

1) A stream filter instance identifier, an integer value that uniquely identifies the filter instance in the ordered list of filters.
2) A stream_handle that can be either an integer value associated with the stream that the packet belongs to or a wildcard.
3) A priority specification that can be either a single priority value or a wildcard. The value of the stream_handle and priority parameters associated with a received frame determine which stream filter corresponds to the frame and, therefore, the filtering and policing actions to be applied to it.
4) The stream gate instance identifier that indicates the stream gate instance to be used.
5) Zero or more filter specifications that determine if a frame passes or fails the specified filter. The frames that fail a filter are discarded. Among the filter specification, there are the maximum service data unit (SDU) size (that discards the frames exceeding the maximum SDU size defined for the stream) and the flowmeter instance identifier that identifies the flow metering function to be applied.
6) Several frame counters that account for the number of frames that pass/fail the stream gate, that pass/fail the maximum SDU size filter, that are discarded by the flowmeter, and so on.

A stream gate has a unique identifier, a state (open/close) that determines whether a frame is allowed to pass through the gate or not, and an internal priority value (IPV) that is used by the Qci protocol to determine the frame traffic class.

Qci provides for quality of service protection, which is very useful when multiple streams share the same switch egress queue, with a potential for interfering with each other. In these circumstances, in fact, if some stream does not comply with the assigned bandwidth, the other streams could experience long latency in the queue and even frame loss in case of overload. To deal with these issues, Qci allows to monitor streams and to detect if some stream does not respect the committed behavior (e.g., exceeding the bandwidth usage or the burst size at the ingress to the bridge). If violations are detected, Qci takes mitigating actions. For instance, there are applications in which frame transmissions and receptions are coordinated and, therefore, frames have to be received only when the stream gate is open. In these cases, a frame received by the stream gate while it is in the closed state indicates the occurrence of an invalid condition, which must be identified and handled. To this aim, Qci provides optional mechanisms (based on the GateClosedDueToInvalidRx parameter of the stream gate instance) that allow to detect any frame incoming during the time intervals in which the stream gate is closed and to permanently set the stream gate to the closed state, thus blocking the future frames of the specific stream until the intervention of the network management.

The execution of the gate operations in a stream GCL is based on the cyclic properties of the Qbv standard and is controlled by three state machines that are specified as the ones used for scheduled traffic, with additional definitions of procedures and variables.

The Qci capabilities for executing a list of timed gate control operations in order, splitting the cycle time into time periods during which the gate is open or closed, and changing the IPV are exploited in the IEEE 802.1Qch–Cyclic Queuing and Forwarding standard.

2) Mechanism Discussion and Open Issues: Qci standardizes a versatile set of knobs and dials to configure synchronized and unsynchronized traffic policing actions. The standard also hints to certain configuration types, e.g., "NOTE 2—The use of stream identifier and priority, along with the wild-carding rules previously stated, allows configuration possibilities that go beyond PSFP as implied by the subclause title; for example, per-priority filtering and policing or per-priority per-ingress port filtering and policing can be configured using these rules." [IEEE 802.1Q-2018, clause 8.6.5.1.1]. However, from the perspective of the authors, the research in the concrete application of Qci is lacking behind the other TSN standards. Thus, guidance on how to concretely configure Qci to achieve a concrete effect is largely missing and subject to future in-depth analysis.

E. Redundant Transmission IEEE 802.1CB

The IEEE 802.1CB standard—Frame Replication and Elimination for Reliability (CB) reduces the packet loss probability by replicating packets, sending them on separate paths and then eliminating the replicas. More specifically, CB enables sequence numbering, replication of every packet in the source end station and/or in relay systems in the network, transmission of the duplicates over separate paths, and their elimination at the destination end system and/or in other relay systems. CB does not cover the creation of the multiple paths over which the duplicates are transmitted (this is realized using other protocols, such as the IEEE Std 802.1Q, IEEE Std 802.1Qca, and IEEE P802.1Qcc).

1) Mechanism Overview: CB assumes a stream as a sequence of packets sent from a talker to one or more listeners either unicast or multicast (i.e., the path can be point to point or point to multipoint).

The CB standard defines schemes for identifying packets belonging to streams and distinguishing them from other packets. Stream identification exploits a single service access point (SAP) to a connectionless packet service realized by the layer below and provides an array of SAPs to the layers above that correspond to different streams. CB exploits the following subparameters.

1) *Stream_handle:* An integer that identifies the stream to which the packet belongs. Such a value is local to a system, so there is no mandatory one-to-one correspondence to an explicit field in the packet.
2) *Sequence_number:* An unsigned integer that indicates the packet transmission order relative to other packets of the same stream. The sequence_number subparameter for a given stream on a particular port and direction can be inserted in and extracted from the packet by either the sequence encode–decode function or the stream identification function.

Stream identification can be active or passive. On the transmission side, active stream identification functions modify the data parameters of the packets received from the upper layers, encoding the information relevant to the SAPs to use, and then forward the encapsulated packets to the lower layers. On the receiver side, active stream identification functions decapsulate the packets received from the lower layers and pass them to the upper layers through the SAPs encoded in the packets. Conversely, passive stream identification functions let the packets received from the upper layers pass unmodified but inspect any incoming packet received from the lower layers in order to identify the stream to which the packet belongs and determine the relevant SAP. CB provides four specific stream identification functions as follows.

1) The null stream identification function, which operates at the frame level and is suitable for applications, such as AVB streams [39] that have a unique {destination MAC address, VLAN} pair per stream.
2) The source MAC and VLAN stream identification function, which also operates at the frame level, is suitable for applications in which all data packets characterized by a given {source MAC address, VLAN} pair belong to the same stream.
3) The active destination MAC and VLAN stream identification function, which also operates at the frame level. This function, within a talker, is suitable for translating a particular stream to use a particular {MAC address, VLAN} pair to make the stream identifiable by the IEEE 802.1Q bridges in the network. Within a listener, the function is used for recovering the original addressing information before passing the packet up the protocol stack. The function is also suitable for a relay system that acts as a proxy for a listener.
4) The Internet protocol (IP) stream identification function, which operates at the transport layer and

Table 2 Stream Identifications and Their Parameters

Stream Ident. Function Name	Type	Inspected parameters	Modified parameters
Null Stream identification	Passive	dest. address, vlan identifier	None
Source MAC and VLAN Stream identification	Passive	source address, vlan identifier	None
Active Destination MAC and VLAN Stream identification	Active	dest. address, vlan identifier	dest. add., vlan id., priority
IP Stream identification	Passive	dest. address, vlan identifier, IP source add., IP dest. add., DSCP, IP next protocol, source port, dest. port	None

IP interface layer. This function can be combined, for instance, with the active destination MAC and VLAN stream identification function to assign a particular {MAC address, VLAN, priority} triple to packets belonging to a particular unicast IP stream.

Table 2 gives the parameters that each of these function inspects and the ones that it modifies, if any.

CB has features that favor interoperability, flexibility, and robustness. A degree of interoperation between IEEE 802.1CB systems and systems based on some other, similar protocols, such as high-availability seamless redundancy (HSR), parallel redundancy protocol (PRP) [40], or RFC 3985 pseudowires [41] can be achieved, thanks to some common properties. For example, the use of 16-bit sequence numbers and the CB ability to be configured to adopt per-source sequence numbering, which is the configuration supported by the IEC 62439-3 HSR/PRP standard.

Flexibility is provided by the CB ability to support various stack positions, which makes the range of applicability of the protocol quite broad. For example, CB functions can be performed in the two end systems (if they are CB-enabled) and in any relay along the path. If an end system has no CB capability, one or more adjacent relay systems serve as proxies for accomplishing CB functions. The network can be configured so as to discard and rereplicate packets at various points along the path in order to tolerate multiple failures and complex configuration are supported.

CB eliminates duplicates on the basis of the sequence numbers carried in the frames. For this, CB maintains a window of valid sequence numbers, delivers frames to higher protocol stacks (or to an application) for which the sequence number in the said window has not yet been delivered, and eliminates received frames with sequence numbers that have already been delivered. The size of the window of accepted sequence numbers is configurable. When the window is configured to be larger than one (which will regularly be the case) and, therefore, multiple sequence numbers are valid at the same time, CB may deliver frames out of order, i.e., CB may deliver frames with later sequence numbers before frames with preceding sequence numbers. While out-of-order delivery of frames may not be ideal for an application, it is a cost-efficient solution for CB. In-order delivery would require the buffering of frames before delivery until all frames with preceding sequence numbers would be delivered. The effects of sequence numbers in frame elimination have been formally analyzed in the context of CB by means of model checking [42].

In order to meet the robustness goal, i.e., to cope with errors due to a stuck transmitter repeatedly sending the same packet, an individual recovery function is defined that removes repeated sequence numbered packets received from the stuck transmitter.

2) Mechanism Discussion and Open Issues: CB targets to improve the reliability of message transmission, where reliability is formally defined as the probability of correct operation from system start until a particular point in time [43]. Therefore, for example, reliability of a network would be the probability that said network delivers its messages (i.e., at least one copy of a message when CB is used) for a duration of a year. As the network will typically be only a part (although a very important one) of an overall system, such as a factory or a car, the improvements in reliability of the network typically improve the availability of the overall system. Availability is formally defined in [43] as MTTF/[MTTF + MTTR], where MTTF is the mean time to failure and MTTR is the mean time to repair. Thus, availability takes the possibility of repair into account, while reliability does not. The improvement in the reliability of the network increases the MTTF and, thereby, the availability of the overall system.

However, in order to achieve the effects of improved network reliability and emerging system availability, CB must be installed with great care. The first, rather straightforward observation targets restrictions on the network topology: in order to avoid any single point of failure in the network, at least two disjoint paths between a talker and its listeners need to be installed. These disjoint paths include the connections from the talker to the network and the connections from the listeners to the network, i.e., an end station must connect to the network by two physical links. Otherwise, if the end station connects by one link only, the failure of the said link will cause a transmission failure that cannot be recovered by CB.

The second, not so obvious observation addresses the composability (actually the noncomposability) of CB with E2E protocols. E2E is often used in safety-critical applications where the network is considered a black channel that is not trusted. For example, it is considered a possible failure behavior of a bridge that a faulty bridge may change the payload of an Ethernet frame and generate a matching Ethernet FCS. E2E is, therefore, executed as additional protocols beyond the Ethernet FCS and is executed as part of an application or as a layer close to the application.

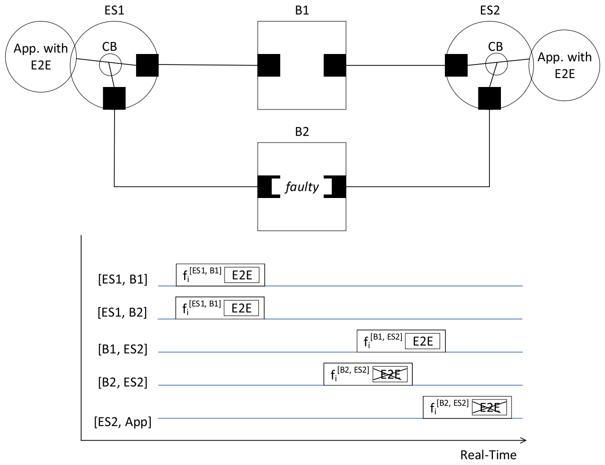


Fig. 8. *Extended network—CB incompatibility with E2E protocols.*

Typical E2E mechanisms are checksums, sequence numbers, and timestamps. The rationale behind a back-channel approach is discussed, for example, by Saltzer *et al.* [44] in their famous paper The E2E Argument.

The E2E paradigm assumes that the network, in particular, in the failure case, is not able to compromise the E2E mechanisms and, therefore, modifications of a message by the network are detected by the listeners. This assumption needs a safety argumentation, and this safety argumentation will always be based on a first element, which says that the network (switches and end stations) does not have knowledge of the E2E mechanisms deployed, and the second element, which says that it is sufficiently improbable that the network by chance will successfully modify a message (or construct a new message) such that this modification/construction is undetected by the listener.

The first element is the root cause that makes E2E protocols not composable with CB. We illustrate the problem by extending the simple network with the second bridge B2 (see Fig. 8). Such a bridge acts as a disjoint path between ES1 and ES2, i.e., in addition to the transmission path of the simple network [[ES1, B1], [B1, ES2]], there is a second transmission path from ES1 to ES2 via B2: [[ES1, B2], [B2, ES2]]. We assume that B2 is faulty. Furthermore, messages ES1 toward ES2 use CB; therefore, they are sent over both paths and, in ES2, CB forward the first received copy of a message to the application and removes the duplicates. We also assume that the transmission via B2 happens to be faster than the transmission via B1. As a result, ES2 receives messages from B2 earlier than messages from B1 and, consequently, CB forward the messages received via B2 to the application. Since per our safety argumentation, the ES2 does not have knowledge about the E2E mechanisms (only the application executing on ES2 has) and CB in ES2 cannot decide whether the E2E mechanisms are correct or not (e.g., CB cannot decide that an E2E checksum is correct because CB does not have knowledge about the polynomial in use). Thus, in the case that, indeed, B2 modified the message such that the E2E mechanisms are corrupted, only the application will find out. Yet, since CB in ES2 will eliminate the corresponding message from B1 because it is received after the message from B2, the application will not receive the copy of the message in which the E2E mechanisms would be correct (as we assume B1 to operate nonfaulty).

Despite the noncomposability of CB with E2E protocols, there is still value in using CB, as there are many industrial systems that do not require E2E protocols but take advantage of improved network reliability. Said industrial systems will, though, not be safety-critical.

F. Configuration IEEE 802.1Qcc

As a network becomes more and more capable, it tends to define more and more knobs and dials that need configuration (some also provide means for network diagnosis and monitoring). Q defines these knobs and dials as managed objects in clause 12 of the standard and TSN introduces configuration models [45] that operate on said objects. We discuss these configuration models in this section.

1) Mechanism Overview: The configuration models define how talkers/listeners may request specific properties for streams they source/consume. Likewise, the models define how bridges may react to said requests with status information, e.g., to indicate whether the network can satisfy the requests and/or how well the requests can be met. The configuration models are models, indeed, in that they define abstract functionality rather than concrete protocols. This abstract functionality consists of the definition of a user-network interface (UNI), the localization of the UNI within the network, and whether the managed objects are modified based on a distributed or a centralized configuration approach. Different configuration models are: the fully distributed model, the centralized network/distributed user model, and the fully centralized model. For the discussion of the models, we will use an extended network that extends the simple network in the following way: instead of a single bridge, the extended network comprises ten bridges B1–B10. ES1 continues to connect to B1, but ES2 now connects to B10. B1 connects to B2 and any other bridge Bi connects to B$i+1$, such that the extended network forms a line topology with the end systems ES1 and ES2 on opposite ends of the line (see Fig. 9). Furthermore, in this extended network, ES1 is a talker and ES2 is a listener of a stream f_i.

In the fully distributed model, the UNI is located between the talker/listener and the bridge they connect to. Hence, in the extended network introduced earlier, the UNI is between ES1 and B1 as well as between ES2 and B10. ES1 may exercise the UNI by advertising its stream f_i to B1. This informs the network that ES1 is ready to publish f_i and also informs the network of communication requirements for f_i (e.g., necessary bandwidth

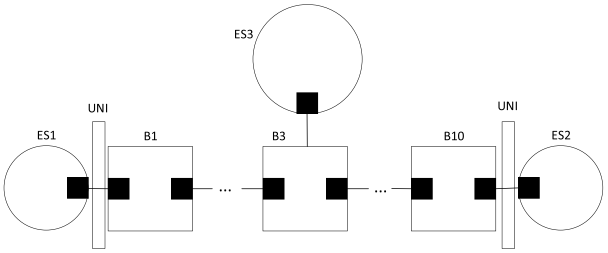


Fig. 9. ***Extended network—Qcc example—fully distributed model.***

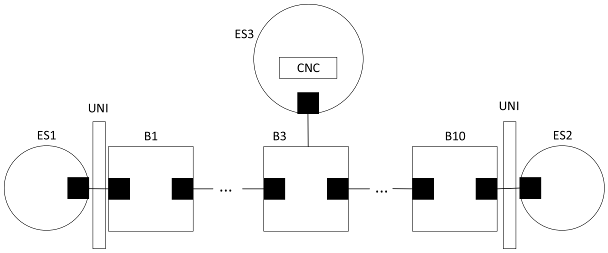


Fig. 10. ***Extended network—Qcc example—centralized network/distributed user model.***

reservations for f_i). B1 propagates this talker advertise along to B2 and so on, until the advertisement reaches a listener, ES2 in the extended network case. Now, a listener (i.e., ES2) may use the UNI to inform the network that it is ready to receive an advertised stream (i.e., f_i from ES1). Again, this listener ready is communicated back to the talker over the network (i.e., toward ES1 along the bridges forming the line topology). In this propagation of the talker advertise and the listener ready, each bridge builds up status information associated with the said stream. Some of this status information will indicate whether the network is actually able to meet the communication requirements for a stream, or not. For example, each bridge may maintain a managed object that keeps track on how much bandwidth there is available for communication between bridges and modify this status when a new stream is successfully configured. For example, in the extended network, f_i may only be configured if all bridges report sufficient bandwidth to be available on the bridge-to-bridge links. On the other hand, if a link is already saturated, e.g., the link between B5 and B6 (because of other communication of other end systems not discussed in the extended network description), the stream may not be configured.

The SRP (Q, clause 35) is an example of the fully distributed model (and, in fact, the emphasized terminology in the paragraph mentioned above is taken from SRP).

It is a defining characteristic of the fully distributed model that each bridge is taking only local decisions on how to update its managed objects, in the absence of a centralized coordinating entity.

In the centralized network/distributed user model such a centralized coordinating entity, the centralized network configuration (CNC) is installed. In the extended network, the CNC may be realized by another end station ES3 connecting to, say, bridge B3 (see Fig. 10).

The UNI remains between the talker/listener and the bridge toward which the end systems connect. However, the configuration requests are not addressed locally by each bridge in the network, but the requests are rather forwarded to the CNC (located in ES3 in the case of the extended network). The CNC generates configurations for all bridges affected by the talker/listener request and provides the configuration to the bridges, while the bridges exercise the UNI to forward configuration information to the talkers/listeners. The CNC has a global view of the resources in use and available in the bridges and can, therefore, generate more efficient configurations than the fully distributed model. Therefore, it will sometimes even be the case that the CNC will find configurations when the distributed model will fail to configure streams.

Finally, the fully centralized model introduces another element: the central user configuration (CUC). In this model, the UNI is located between the CUC and the CNC rather than directly at the talker/listeners. Thus, the end stations coordinate their communication needs with the CUC and it is the CUC that requests configuration updates from the CNC on behalf of the talker/listeners. The CNC then produces a new configuration, distributes the configuration to the bridges in the network, and returns status information to the CUC. The CUC completes the coordination with the talker/listeners by forwarding relevant configuration information from the CNC to the end stations. In an extreme case, the CUC has full knowledge of the communication requirements already at design time of the system (e.g., an operator provides a configuration file to the CUC) and talker/listeners are only receivers of configuration information that the CUC negotiates with the CNC.

In the extended network, the CUC may be implemented in ES3 side by side with the CNC (see Fig. 11). End stations may use a protocol outside TSN to communicate with the CUC, for example, OPC UA [46] (basic Ethernet connectivity is assumed to be available between end stations and the CUC). ES1 registers f_i at CUC and ES2 may request f_i from the CUC.

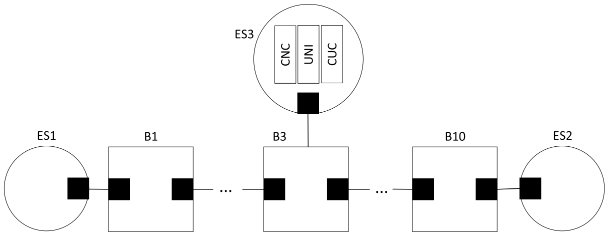


Fig. 11. ***Extended network—Qcc example—fully centralized network.***

In this model, the following configuration actions may be executed.

1) ES1 registers at the CUC that it is able and willing to transmit f_i.
2) ES2 requests from the CUC to receive f_i.
3) The CUC exercises the UNI to the CNC with the parameters of f_i, thereby asking the CNC to setup the network in such a way that the transmission of f_i from ES1 to ES2 is guaranteed with the desired communication properties (e.g., real-time properties).
4) The CNC configures the bridges B1–B10.
5) The CNC provides the configuration information for ES1 and ES2 to the CUC (e.g., which stream identifier the end systems shall use).
6) The CUC sends the configuration information to the end stations.

In addition to the generic definition of the UNI, TSN also provides a concrete realization of the UNI for the centralized models. This concrete realization is formulated in the data modeling language YANG [47]. These YANG models may be communicated between the talker/listeners or the CUC and the CNC by protocols like Netconf or Restconf (although TSN does not mandate a specific protocol for the communication of the YANG models).

2) Mechanism Discussion and Open Issues: The choice of configuration model demands a careful analysis of the network's application domain, but we can formulate generic observations.

A first observation targets algorithmic complexity: centralized algorithms tend to be significantly less complex than distributed algorithms because race conditions do not occur. For example, in scenarios of concurrent stream reservations and the fully distributed model, some bridges may reserve resources for a first stream, while other bridges reserve resources for a second stream. If the resources allow only one more stream reservation, then the network will not be able to reserve either one of the two streams. Distributed algorithms that deterministically resolve such situations are algorithmically complex and, therefore, error-prone—there is a good chance that corner cases are overlooked. In the centralized models, these problems are avoided by design, since there is only one deciding entity (i.e., the CNC), either one of the streams will be reserved.

A second observation is on the execution time of algorithms: the centralized models may impose a temporal overhead over a distributed solution, as the reconfiguration requests and responses need to be communicated to and from a centralized instance. On the other hand, in the distributed model, algorithms can be designed, in principle, in which the talker, the listeners, as well as only the interconnecting bridges between the talker and listeners are involved (in the centralized model, the CNC may be located in a different network segment outside the communication paths between the talker and the listeners). The communication speed as well as the frequency of reconfiguration attempts are two key parameters that allow the calculation of said temporal overhead, and we assume that the communication speed is sufficiently high and the reconfiguration frequency is sufficiently low, such that the temporal overhead of the centralized configuration models is acceptable for most industrial use cases.

A third observation is that centralized models tend to cause bottlenecks and other performance limitations both from the computation as well as from the communication perspective. Some industries, therefore, deploy dedicated networking infrastructures for network configuration and management. This so-called out-of-band management provides full isolation of user data from management data. Out-of-band management is useful, for example, to ensure that hostile user data cannot maliciously change the network configuration. On the other hand, the out-of-band management is rather costly because of the additional cost of the separate infrastructure. In most cyber-physical systems (CPS), such as industrial automation and automotive, separate networks are installed today primarily to achieve freedom of interference of safety-critical from nonsafety-critical systems. The configuration of such systems is then either done once at design time or through the same network that carries user data, i.e., in-band. TSN also targets to integrate safety-critical applications and nonsafety-critical applications on a single physical infrastructure. Again, we assume that most industrial use cases will provide sufficiently high bandwidth and sufficiently high computing resources so that bottlenecks and performance limitations of the centralized model can be avoided.

A fourth observation is in the area of fault tolerance: centralized solutions bear the risk of a single point of failure—if the CNC becomes faulty, reconfiguration is not possible anymore. However, also distributed models are not by default fault-tolerant but must be designed to tolerate failures. Analogously, replicated CNCs can be installed in a network and well-known replication strategies such as hot/warm/cold standby can be deployed. On the other hand, the unavailability of the CNC may actually not be a time-critical scenario and manual maintenance may be deployed as corrective action.

A fifth observation is on the solution efficiency: we already discussed earlier in this section that the centralized models may find better solutions than the distributed model or may even find solutions where distributed models would fail since the centralized models have more information available and can search for global optima. Centralized models may also be an easier platform to realize self-configuration/self-optimization approaches of the network, e.g., Gutiérrez *et al.* [48] refer to the CNC as an ideal location to implement a configuration agent.

A sixth observation is cost efficiency: while a detailed cost comparison between the centralized and the distributed models is outside the scope of this paper, we want to emphasize that an argument that calls out the centralized solution to be less cost-effective solely on grounds of a requirement for additional components (e.g., CNC and CUC) is likely not sustainable. Risks of algorithmic complexity, development time, performance, and solution efficiency must be factored in when conducting a detailed cost-efficiency comparison. In some use cases, the CNC (and the CUC) may even not be present as part of the network in operation at all but may only occasionally be connected to the network, e.g., as a cloud-based solution.

In the context of these reflections, the rise of software-defined networking (SDN) solutions, for example, in the information technologies (IT) and telecom domains do not come as a surprise. According to SDN, the network distinguishes between the control plane from the data plane, where the control plane configures how messages are transmitted through the network, while the data plane is the actual transmission process. Furthermore, a central network controller is installed to centrally manage the control plane. SDN in the IT and telecom domains is often realized by Openflow [49], a concrete set of protocols and techniques. While Openflow is, in principle, also applicable to TSN, Netconf, Restconf, and YANG are more likely building blocks to realize SDN in TSN [50]. The relation of SDN to TSN and their integration is an active field of research [48], [51]–[53].

Indeed, the authors of this paper believe that centralized configuration will address most industrial use cases and should be considered the default configuration model of TSN.

G. Other TSN and Related Standards

This section briefly discusses other TSN and related standards. Some of them, at the time of writing, are still at an early stage. Other ones are only partially relevant to the industrial automation and automotive domains, while others do have a potential for being adopted in these areas, but their acceptance has not been sufficiently demonstrated so far.

1) IEEE 802.1Qca: Qca (Path Control and Reservation) extends the IS-IS protocol allowing for the establishment of explicit trees to be used for placing selected traffic on a precisely defined route, other than the shortest path tree, to improve resiliency and decrease the probability of congestion. Qca also provides mechanisms for bandwidth allocation and improves redundancy in many ways, e.g., through protection schemes based on multiple redundant trees, local protection for unicast data flows based on loop-free alternates, and restoration after topology changes upon a failure event. The adoption of Qca in the industrial automation and automotive areas is unclear since the respective networks in these fields are often engineered (i.e., preplanned), especially in safety-relevant or safety-critical application cases.

2) IEEE 802.1Qcr: Qcr standardizes mechanisms for guaranteed low-latency transmissions without the need of a synchronized timebase. For this, Qcr standardizes Asynchronous Traffic Shaping, an additional layer of shaped egress queues based on the concepts introduced in [11]. Qcr is likely to become a relevant TSN mechanism for the automotive industry as indicated by the contributions to this standardization project.

3) IEEE 802.1Qch: Qch standardizes Cyclic Queuing and Forwarding which enables a bridge to define the egress queue of a frame based on its arrival point in time. One use case of Qch aims to minimize the relay jitter of a frame in a switch. This is achieved by the switch maintaining at least two egress queues per port that are mutually exclusively enabled and disabled according to a period schedule (e.g., by using Qbv). Thus, at any time, only one of the two queues is enabled for transmission. While one queue is enabled, all received messages during this time are allocated to the respective other queues (which is disabled). When the two queues toggle their states, all messages in the previously disabled queue are now forwarded and newly received frames are allocated to the respective other (now disabled) queue. Because of the resulting traffic shape, this method has originally been named peristaltic shaper. At the time of this writing, there are no strong indications that Qch will be adopted in industrial application areas.

4) IEEE 802.1CS: CS defines a Link-Local Registration Protocol that allows to replicate a registration database between two directly connected devices. This replication targets data with a size of up to 1 MB. The Link-local Registration Protocol is a building block of the ongoing IEEE 802.1Qdd standardization project (see next).

5) IEEE 802.1Qdd: Qdd aims to standardize a Resource Allocation Protocol. RAP will be a distributed resource management and admission control protocol for networks with latency and bandwidth guarantees. RAP will complement the multiple SRP (MSRP) that has limitations in terms of the number of reservations, admissions, and configuration size in general. Qdd has a good potential to become relevant for the industrial automation area. As the standardization project has been started rather recently, a detailed analysis at the time of this writing is not possible.

6) IETF Detnet: The IETF maintains a working group on deterministic networks called detnet. While TSN is focusing mostly on Layer 2 networks (with some exceptions, like ASrev targeting also higher layers), IETF standardizes functionality on Layer 3 and higher. Thus, especially large-scale networks may benefit of a combined TSN/detnet architecture that would guarantee deterministic stream transmission "E2E" even in case the stream needs to cross routers. While detnet is an attractive approach, many areas of CPS, such as industrial automation and automotive, have existing solutions ready for Layer 3 and above. For example, automotive has Adaptive AUTOSAR communication primitives (i.e., SOME/IP) and industrial automation is planning to adopt OPC/UA. Thus, although detnet would in principle be suitable as standards-harmonization project complementing TSN, at the time of this writing, it is still unclear to what extent detnet will evolve and become accepted in these industries.

V. USE CASES

Although TSN is a rather novel technology, multiple use cases in various industries are lining up already. We briefly discuss some of the uses cases next and will then focus on industrial automation and automotive use cases.

Simon *et al.* [54] discuss a use case in the telecommunication domain where TSN is to be used for fronthaul networks. Fronthaul networks are rather large in size, e.g., 1-km physical links and have high capacity, e.g., 10 Gb/s. They transport radio data that are conformant to the CPRI [55]. In their studies, Simon *et al.* [54] demonstrate the beneficial effects of Qbv and Qbu on communication in fronthaul networks and discuss an additional mechanism, called playout buffer, which can be realized together with TSN to achieve even better real-time performance. Hence, TSN is suitable to improve and support telecom networks and, in particular, can become a key element of the upcoming 5G infrastructure. On the other hand, as 5G also targets the market of CPS itself, e.g., industrial automation, recent research investigates also the integration of 5G and TSN [56].

Jakovljevic *et al.* [57] discuss requirements on future in-train communication networks and outline novel system architectures. It is suggested to base the new architectures on TSN and, potentially, to complement TSN with elements from SAE AS6802 [58] and ARINC 664 [59].

Steiner *et al.* [60] present a use case in aerospace where TSN is used as cabin backbone bus (CBS). CBS has stringent requirements for cabin announcements like for boarding the aircraft or in critical situations. Qbv, AS, and CB are mentioned as being beneficial to the CBS design.

Today, the duty vehicle and agriculture vehicle/machine industry deploys a variation of the CAN bus, called ISOBUS. This industry is currently investigating alternatives for a high-speed ISOBUS for which TSN is a promising candidate.

Concrete use cases for AD are presented in a recent publication by Samii and Zinner [10]. Here, the authors provide example network architectures that demonstrate the benefits of TSN to the automotive industry. We will discuss automotive use cases in more detail in the following.

It is the authors' perspective that the industrial automation industry as well as the automotive industry will be the first to adopt TSN as their mainstream means of networking. Indeed, at the time of this writing, a consortium of major suppliers in the industrial automation industry has publicly announced to champion TSN together with OPC UA as their future mainstream communication solution.[4] This marks an important milestone that even caused key proponents in industrial automation to proclaim: "the (fieldbus) war is over!".[5] Furthermore, a joint standardization project of IEC SC65C/MT9 and IEEE 802 has been established: IEC/IEEE 60802 develops a TSN Profile for Industrial Automation. We will discuss this activity in more detail in the following section.

While the industrial automation industry is about to adopt TSN (and OPC UA) as their new mainstream means of communication, the automotive industry is not as advanced. One of the key reasons of delayed broad automotive adoption of TSN may be because the automotive industry only rather recently started to implement Ethernet as means of in-vehicle communication, while industrial automation has been using various real-time Ethernet networks for decades. However, there are many signs indicating that Ethernet and TSN will also become a mainstream network for in-vehicle communication. First, various automotive companies participate in the IEEE 802.1 TSN task group. Second, over the last couple of years, the automotive industry has organized several conferences on TSN with stable and growing automotive industry participation. Third, automotive-grade TSN solutions are already available on the market. Fourth, the authors note an increase in exchange and research projects with automotive companies, which indicates that TSN standards are likely to be adopted in future cars.

A. Industrial Automation Use Cases

The building blocks of smart factories and Industry 4.0 are CPS, which typically include controlling devices (e.g., PLC), sensors, actuators, drives, human–machine interface (HMI), interface to the upper level (e.g., with PLC acting as gateway and/or router and/or bridge) as well as other Ethernet devices, such as servers, and diagnostic equipment. The advent of Industry 4.0 opens up new manufacturing scenarios, which embed and integrate modern technologies, such as advanced robotics, artificial intelligence, sophisticated sensors, cloud computing, and big data analytics to increase productivity, flexibility, and product quality. In these scenarios, the physical and virtual worlds will interlink. Consequently, the typical CPS that monitor the physical processes in charge of automated manufacturing management, often referred as operational technologies (OT), will interoperate with IT, such as cloud computing and service-oriented architectures, as it is shown in Fig. 12.

While OT is designed to fulfill properties, such as real-time and safety-critical behavior, reliability, and availability, IT is usually not designed with those properties in mind. Consequently, the challenge is to make OT and IT coexist. One notable example is communication. In fact, the significant increase in the demand for networking and the availability of high-speed Ethernet equipment, which nowadays is cheaper than the one of special-purpose digital technologies, result in critical traffic flows

[4] https://opcfoundation.org/news/press-releases/major-automation-industry-players-join-opc-ua-including-tsn-initiative/

[5] https://www.linkedin.com/feed/update/urn:li:activity:6473453300460056576/

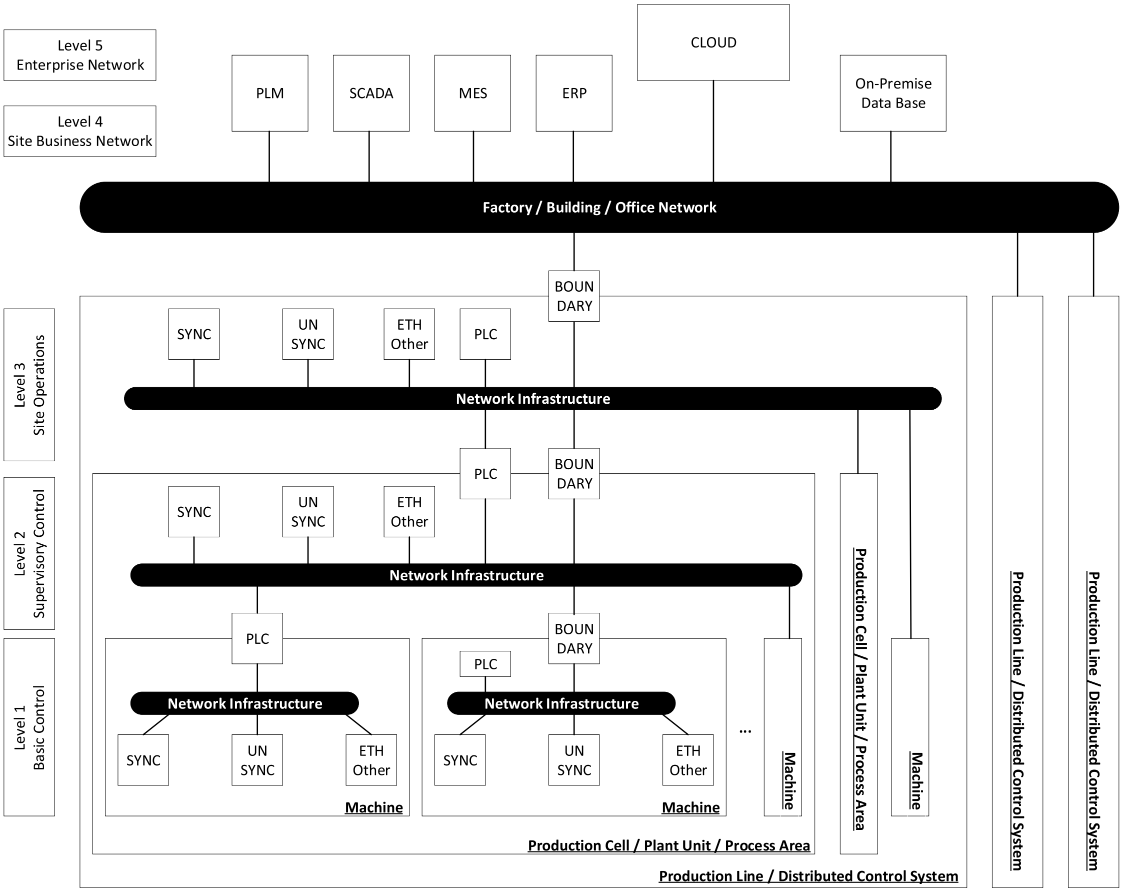


Fig. 12. *Hierarchical structure of industrial automation (from [62]).*

(e.g., time-sensitive ones) and noncritical traffic flows sharing the same network. TSN is the foundation to provide connectivity to time and mission-critical applications over converged Ethernet networks. With TSN, a network can consist of multiple-vendor devices that can interwork and can be configured via a single standard interface. However, in order to deploy converged networks able to simultaneously support OT traffic and IT traffic, developers, vendors, and users of interoperable bridged time-sensitive networks for industrial automation need guidelines for selecting features, configurations, protocols, and procedures of bridges, end stations, and LANs. The answer to this need is the ongoing standard named IEC/IEEE 60802–TSN Profile for Industrial Automation (TSN-IA), which is a joint project of IEC SC65C/MT9 and IEEE 802.

The IEC/IEEE 60802 standard defines profiles for network bridges and end stations for TSN in industrial automation, which are based on standards published by IEEE 802.3 and IEEE 802.1. The choice of using the IEEE 802.3 and IEEE 802.1 standards as the building blocks for the lower communication stack layers and the management is to avoid the proliferation of divergent implementations of deterministic real-time Ethernet networks while exploiting the advantages provided by Ethernet networks in terms of deterministic transmission, data rate availability (from 10 Mb/s to 1 Tb/s), time synchronization, and so on.

The TSN-IA selection of IEEE standards includes: the IEEE 802.3-2018–IEEE Standard for Ethernet, the IEEE 802.1Q-2018—Bridges and Bridged Networks, the IEEE 802.1AB-2016—Station and Media Access Control Connectivity Discovery, the IEEE802.1AS-rev—Timing and Synchronization for Time-Sensitive Applications (whose publication is expected in 2019) and, optionally, other ones, such as the IEEE802.1CB-2017—Frame Replication and Elimination for Reliability and the IEEE 802.1X-2010—Port-based Network Access Control. In addition, the TSN-IA profiles shall support the coexistence of different data streams between end devices, including the communication profile families (CPF) defined in the IEC 61784-2 standard, in order to cover the broad spectrum of requirements imposed by the diverse IA applications. The aim of the TSN-IA standard is to pursue interoperability of bridges and end stations in the industrial automation domain as far as stream configuration and establishment, and network configuration are concerned. The selected TSN-IA profiles cover both the IEEE 802-defined layer 2 and the protocols to configure layer 2.

At the time of writing, the IEC/IEEE 60802 standard is in progress.[6] Some documents that deal with the requirements that the IEC/IEEE 60802 standard has to cover are publicly available. In particular, one of the main sources of requirements for the ongoing standard is the document [61], which is based on the industrial use cases described in [62]. Such a document identifies different traffic schemes/patterns that IA applications generate for implementing different functionalities relevant to parameterization, control, or alarming. Among the listed traffic types, there are isochronous cyclic real-time, cyclic real-time, network control, audio/video, alarms/events, configuration/diagnostics, internal/pass-through, best-effort, and brownfield traffic (i.e., generated by devices that are not conformant to the TSN-IA). Reference [62] describes the diverse characteristics (i.e., periodic/sporadic, guarantee, data size, redundancy, etc.) of these traffic schemes/patterns and the different requirements that they impose on a TSN network, and presents relevant use cases. For each use case, properties and requirements are provided, often with numerical values, and the TSN mechanisms that are suitable for dealing with them are identified.

A central concept in the use cases [62] is the TSN domain, which is defined as a number of commonly managed industrial automation devices whose grouping is the result of administrative decisions. In other words, a TSN domain consists of a set of end stations and/or bridges, their ports and the attached individual LANs that transmit TSN streams using TSN standards, and that share a common management mechanism. For example, typically machines are separate TSN domains, production cells, and lines can also be set up as TSN domains, while devices can belong to multiple TSN domains at the same time. TSN domains may be connected by bridges (level 2), routers (level 3) or application gateways (level 7). In addition, wireless access points and 5G base stations may be used too.

As is shown in Fig. 13, multiple overlapping TSN domains are originated when controllers use a single interface for the machine-to-machine (M2M) communication with controllers of the cell, line, plant or other machines. For example, considering machine 1, the controller link to the cell bridge B1 is a member of the TSN Domains of machine 1, production cell 1, production line, and Plant. In the case of machine 3, instead, the controller is directly attached to the PLC of production cell 2 and, therefore, it belongs to the TSN domain of production cell 2, whereas the machine internal TSN Domain is decoupled from M2M traffic by a separate interface.

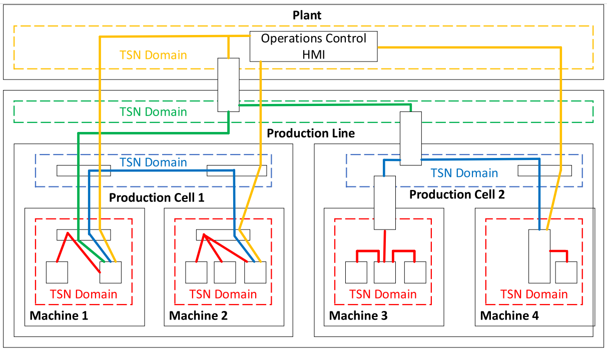


Fig. 13. ***Overlapping TSN domains in the M2M/CTC use case (from [62]).***

As it was mentioned before, a number of use cases to be covered by the IEC/IEEE 60802—TSN profile for IA have been already identified and they are addressed in [62], which provides for each of them a description, the relevant requirements, and the appropriate TSN mechanisms to use. The first use case, the sequence of events (SOE), is relevant to synchronization. It is a mechanism that records timestamped application-defined events (e.g., changes of digital input signal values) from all over a plant in a common database. The requirements for effective SOE identified in [62] are multiple and are relevant to plant-wide high-precision Universal time synchronization, maximum deviation to the grandmaster time (that must be in the range from 1 to 100 μs) and redundancy. The suitable TSN mechanism indicated in [62] is the IEEE 802.1ASrev standard.

In [62], more complex examples can be found. One of them is the M2M/controller-to-controller (M2M/CTC) use case shown in Fig. 13. Here, the various preconfigured machines (machines 1–4) communicate with each other, with the supervisory PLC of the production cell to which they belong (i.e., production cells 1 or 2) or production line or with the operations control HMI. All of them have their own TSN domains. Reference [62] lists the requirements of the M2M/C2C use case. For example, a requirement is that machine internal communication has to be decoupled and protected against the additional M2M traffic and vice-versa. Other requirements are about the support for both one-to-one and one-to-many communications and on the need for a suitable schedule for interleaved operation with machine intervals. Finally, [62] lists the appropriate TSN mechanisms to adopt, among which, 802.1Qbu, 802.1Qbv, 802.1Qci, Fixed priority, 802.3br, Priority Regeneration (i.e., IEEE 802.1Q-2014 clause 6.9.4 Regenerating priority), queue-based resource allocation, and VLANs to separate TSN domains. As the IEC/IEEE 60802 standard writing is in progress, more details and updates can be found on the official website.[7]

A discussion on IA use cases cannot avoid to mention fog computing. In fact, fog computing is a promising

[6] Updated details can be found on the official website https://1.ieee802.org/tsn/iec-ieee-60802-tsn-profile-for-industrial-automation/

[7] https://1.ieee802.org/tsn/iec-ieee-60802-tsn-profile-for-industrial-automation/

option to realize the IT/OT convergence in IA, as it allows to extend switches and routers with computational and storage resources to enable a broad spectrum of communication and computation options. In this context, TSN represents an interesting option for the networking layer of fog computing in IA. However, there are several configuration challenges to face, including the need for run-time configuration ability. On this topic, the work in [63] proposes a configuration agent architecture, based on IEEE 802.1Qcc and OPC Unified Architecture (OPC UA), and a heuristic scheduling approach to perform the reconfiguration of schedules for time-sensitive traffic at run-time. Experimental results proved the feasibility of the proposed architecture and scheduling approach. The usage of TSN for fog computing is subject of the research project Fog Computing for Robotics and Industrial Automation (FORA)[8] which is a Marie Sklodowska-Curie Innovative Training Network funded by the European Commission and trains 15 Ph.D. students on key fog-related technologies. TSN has been identified as one of these key technologies and will be a major item in an upcoming fog computing curriculum that FORA will establish.

B. Automotive In-Vehicle Networking Use Cases

Modern cars implement a variety of electronics functions, which are typically packaged into so-called electric control units (ECUs). Functions need to communicate within an ECU as well as between different ECUs. Furthermore, many functions communicate with the environment that they perceive through sensors and control by actuators. The communication needs are quite diverse for different functions and, therefore, various automotive-specific network solutions have been developed over the past decades. Typically, a car will implement a multitude of those automotive network solutions, examples of common ones are: CAN, LIN, MOST, and FlexRay. More recently, also Ethernet has become available for automotive usage and there are strong indications that Ethernet adoption will be fast and significant [64], [65]. First, TSN enables the usage of a single physical network for applications with different communication requirements—thus, Ethernet can replace multiple networking solutions [66]. Second, as more and more advanced driver assistance systems (ADAS) and AD systems become available, the requirements on the network grow more and more as well. We discuss three automotive use cases next: TSN as a communication backbone between ECUs, TSN as I/O sensor bus, and intra-ECU deployments of TSN.

Many automotive Original Equipment Manufacturers are currently reorganizing the electronic/electrotechnical architecture (E/E architecture) of their vehicles toward a domain-based approach. Example domains are: body controls, energy, gateway/connectivity, infotainment/digital cockpit, chassis and powertrain, and ADAS/Autonomous Driving. Each of the domain groups a number of ECUs and one ECU per said group (typically one of the more capable ECUs) operates as the domain controller. The domain controllers connect to the ECUs of their respective domain, as well as to other domain controllers in the car, for example to the gateway ECU for connectivity to the outside of the vehicle. The connection between the domain controllers is often referred to as the communication backbone where the domain controllers operate also as network switches: communication between any two ECUs from different domains is executed via the network of domain controllers. The realization of the communication backbone by TSN is attractive for the automotive industry since it enables the backbone usage for all applications. For example, time-critical messages can be transported using the Qbv mechanisms, or similar ones, such as the AVB-ST, proposed by Alderisi *et al.* [67] and analyzed in [68], which guarantees E2E latency and jitter. Also, ECUs can transmit safety-critical messages over redundant communication links. The CB mechanism may be used for the redundancy management of medium critical messages. For highly critical messages, the redundancy management may not be executed by CB but only at the application level (so not to compromise E2E safety protocols). Furthermore, provided that there is sufficient bandwidth available on the backbone, also medium or lower criticality applications will be served. The backbone may also offer the capability of protection against babbling ECUs or similar faulty behavior by implementing appropriate traffic policing mechanisms, such as those defined by Qci. This way, it is guaranteed that a faulty ECU from one domain cannot render the backbone unavailable for communication from other domains. Finally, ECUs from within a single domain and between domains can be synchronized to each other by AS. This synchronization may be used directly for communication, like with Qbv, to globally coordinate the execution of tasks, and/or to operate as a timestamping service for critical events.

The number of sensors in a car is also rapidly increasing because of new ADAS and AD functions. Typical sensors in these areas are: cameras (mono and/or stereo), ultrasonic, LIDAR, and RADAR. In particular, for AD major aspects with respect to sensory input need still to be addressed, e.g., what would constitute a necessary and sufficient set of sensors, or whether backup systems may share the same set of sensors as the primary systems. Furthermore, as the car becomes more and more software-defined, it is likely that the assignment of sensors to ECUs will change over the lifetime of the car. Thus, there is a growing requirement for a flexible interconnect between the sensors and the ECUs, which is often called a sensor bus. TSN meets the requirements of a sensor bus very well. Again, Qbv, CB, and Qci may be deployed for time-critical, high-available, and safety-critical communication features. Furthermore, the flexibility can be achieved by means of the centralized configuration models defined by Qcc. For example, one of the domain controllers may operate as CNC to reconfigure the network as necessary. Since not all

[8] http://www.fora-etn.eu/

sensors may have an Ethernet connection, it is likely that gateway devices will find usage much more frequently that translates sensor-specific outputs to the TSN sensor bus. These gateways may operate also as data concentrator: they can bundle multiple sensor outputs and prepare a combined payload for an Ethernet frame.

Even inside an ECU, there are good reasons for deployment of TSN: ECUs implement more and more chips (CPUs, graphics processing units, neural processing units, etc.) and these chips need to communicate with each other. For example, RazorMotion [69] is such an ECU and it deploys a deterministic Ethernet to interconnect multiple chips on the same printed circuit board (PCB). Future ECUs may even have a modular form factor in which multiple PCBs interconnect with an intra-ECU backbone network. Again, TSN can be deployed for this purpose. In both cases, TSN guarantees timely and robust communication between chips.

VI. TSN GENERAL DISCUSSION AND OUTLOOK TO TSN-RELATED RESEARCH DIRECTIONS

While we reviewed TSN mechanism by mechanism in the previous sections, we will now discuss TSN as a whole from the perspective of the authors. We assess TSN with respect to some key categories first and argue promising research directions in the following.

A. TSN General Review

In industrial communication and automation systems (and generally in CPS), the following categories are of upmost importance: communication bandwidth, real-time performance, dependability/fault-tolerance (in particular integrity and reliability), security (in particular integrity, availability, and confidentiality), flexibility, scalability, as well as configuration ease.

1) Communication Bandwidth: As TSN is the evolution of standard Ethernet, it takes full advantage of the Ethernet ecosystem with its ever-increasing bandwidth solutions. Today, for the CPS market, the Ethernet physical layers are already standardized by the IEEE up to 1 Gb/s. Products that implement these standards are already commercially available. Upcoming automotive systems will have a need for even higher communication bandwidth. Consequently, appropriate physical layers for TSN are under development. On the other hand, as TSN is agnostic to the Ethernet physical layer, it can also be deployed at very high link speeds, such as 40 or 100 Gb/s. This makes TSN attractive for use cases even beyond CPS, such as data-center applications.

2) Real-Time Communication: Real-time communication performance has been the prime objective of TSN and, hence, TSN excels in this category. While AVB already introduced real-time communication enhancements with Qav, TSN further improves the performance by means of frame preemption (Qbu, 802.3br), scheduled traffic (Qbv), as well as by asynchronous traffic shaping (Qcr).

However, TSN has not yet reached its theoretical minimum in terms of communication latency, as it assumes store-and-forward switch architectures. Existing industrial Ethernet solutions on the market feature cutthrough communication technologies with on-the-fly message assembly (i.e., nodes/switches connected in daisy-chain topologies update the Ethernet payload while they forward the Ethernet frame). Adding cutthrough is certainly a possibility also in TSN and actually a matter of ongoing debate in the standardization committee. There are also significant arguments against cutthrough adoption, for example: 1) since the transmission of a frame may start before the frame's checksum is received and checked, a bridge may forward a faulty frame; 2) the configuration complexity increases; and 3) the broad applicability of the cutthrough feature is questionable.

On-the-fly message assembly, on the other hand, does not fit the IEEE 802.1 communication model very well as; according to IEEE 802.1, frames may only be modified in a switch according to a limited set of rules. The free modification of the payload of an Ethernet frame is clearly beyond said rules, today. Thus, such a mechanism, although realizable in harmony with TSN, is likely to remain outside of the TSN body of standards.

Another aspect in real-time communication is the jitter of the real-time transmissions. The jitter is the difference between the minimum and maximum latencies of a transmission of a frame. With Qbv (and Qch) TSN implements TT communication and, therefore, full control of the temporal behavior of the transmission. In TT communication, the jitter can be minimized to become as small as the quality (i.e., the precision) of the clock synchronization protocol in place (i.e., AS or ASrev in case of TSN).

3) Reliability: We already formally introduced reliability as the probability of correct operation within a time interval (e.g., from system start to a given point in time in the future). CB is TSN's method to improve reliability of message transmission in the case of message loss, e.g., in cases of link failure, message checksum failures, or fail-silent bridge failures. With Qci, TSN provides methods to further improve the reliability of message transmissions in case of more severe component failures, e.g., babbling idiot failures. However, we also discussed shortcomings in reliability when CB is used together with E2E safety mechanisms.

TSN's clock synchronization protocols (AS, ASrev) also face some limitations when exposed to certain failure models, like asymmetric failures. In such failure cases, end stations and/or bridges may lose synchronized time undetected. This may cause an end station to transmit messages at the wrong time or bridges to operate the Qbv gates at wrong instances in time. Thus, messages may get lost—even when CB is in use. Also, AS and ASrev do not standardize how to merge the timing information from redundant grandmaster clocks. Furthermore, these standards do not describe fault-tolerant network-wide startup nor restart protocols. Thus, it is not guaranteed that a TSN system will become synchronized within an upper bound in time (in certain failure cases) nor is it guaranteed that a TSN network will recover from certain failure cases.

4) Integrity: In TSN, the integrity of an Ethernet message is protected only by means of the Ethernet CRC, which is intended to catch transmission failures on the communication link (e.g., because of electromagnetic interference). From the bit error rate (BER), which is specific to the physical properties of the communication link itself and the operational environment of the network, and the Ethernet checksum the probability follows that a transmission failure remains undetected (i.e., the integrity of an Ethernet frame is violated).

TSN does not provide additional mechanisms to improve integrity. Note that, as previously discussed, CB does not improve integrity: in the above-mentioned scenario, the faulty frame would be provided to the application in case it is the first redundant copy received.

There exists a multitude of (non-IEEE standardized) E2E safety protocols for Ethernet. They typically implement extended checksums, timestamps, sequence numbers, and/or cryptographic signatures. Integrity can also be improved by specific design of the end station and bridges, for example as self-checking pairs.

5) Availability: We have already formally defined availability as MTTF/MTTF + MTTR, i.e., the ratio of the MTTF and the sum of MTTF plus MTTR. Since reliability directly follows from the MTTF, all TSN mechanisms that improve the reliability of the network also improve the availability (i.e., CB, Qci, ASrev). While TSN does not explicitly define repair procedures IEEE 802.1, in general, has been defined for plug-and-play and such features will likely also ease certain repair actions.

6) Confidentiality: Confidentiality of a message is typically achieved by cryptographic means, i.e., the plain text of a message is transformed into a ciphertext representing the message. TSN itself does not standardize methods to improve confidentiality. However, there is a close cooperation of the IEEE 802.1 TSN Task Group with the IEEE 802.1 Security Task Group to investigate how existing IEEE security mechanisms can also be leveraged for TSN networks.

7) Flexibility: TSN defines various network management mechanisms for online reconfiguration, which we already discussed in the context of Qcc. As we have argued for the fully centralized model, the flexibility of the network depends on the capabilities of the central network configuration (CNC) that produces new network configurations based on applications requests. While this approach provides a maximum of flexibility in network reconfiguration, there is no ongoing research (as to the knowledge of the authors) that addresses the worst case time of configuration generation. Thus, applications that require network re-configuration within a certain deadline cannot be reliably served by TSN today. Yet, such use cases appear rather rare for CPS.

Table 3 TSN General Review Summary (From the Authors' Perspective)

Characteristic	TSN	Score
Communication Bandwidth	Eth heritage	* * * * *
Real-Time (latency)	Qbv, Qch, Qbu, Qcr	* * * *
Real-Time (jitter)	Qbv, Qch	* * * * *
Reliability	CB, Qci, ASrev	* * *
Integrity	Eth only	* *
Availability	CB, Qci, ASrev	* * *
Confidentiality	-	*
Flexibility	Qcc	* * * *
Scalability (no. of components)	Eth heritage	* * * * *
Configuration Ease	Eth/AVB heritage	* *

8) Scalability: Ethernet is highly scalable in terms of end stations and bridges. The namespace is rather huge with 2^{48} possible addresses (i.e., identifiers of Ethernet MAC ports). However, there are emerging concerns that even this namespace may reach its limits in the coming years because of developments like the Internet of Things (IoT).[9]

On the other hand, in IT, once Ethernet networks reach a certain size, they are organized in a hierarchical manner. Then, multiple-layer 2 Ethernet networks are connected to each other by layer 3 (and higher layer) routers. Such an approach to structure very large networks is also applicable to TSN networks. However, routers would need to support the TSN mechanisms as well. Indeed, multiple TSN standards (such as ASrev and CB) are already intended also for network layers beyond layer 2.

Of course, the larger the network, the higher the transmission latencies that may occur for individual messages, as each bridge in the message transmission path will add to the message's latency. Cutthrough is a mechanism to shorten the bridge-local latency. However, as discussed, IEEE 802.1 TSN does not address cutthrough message forwarding for now (although this is a matter of ongoing debate).

Also, the quality of clock synchronization in a TSN network is influenced by the network size. Gutiérrez *et al.* [14] have analyzed the degradation of the clock synchronization quality with respect to the network size. They conclude that even in a network with up to a hundred bridge-hops, the clocks remain synchronized below 2 μs with very high probability (no failure cases have been analyzed in this paper).

With Qbv, TSN allows a TT paradigm. As such, it is based on a communication schedule, which even in most simple cases is an NP-complete problem. Thus, the scalability of TSN with respect to configuration of Qbv does not scale very well. However, scheduling is an active field of research and recent schedulers manage to schedule thousands of messages.

[9] https://blog.michaelfmcnamara.com/2013/03/are-we-running-out-of-mac-addresses/

9) Configuration Ease: IEEE 802.1 TSN is a set of generic industry standards for cross-industry usage. As such, these standards tend to implement vast numbers of configuration knobs and dials to maximize their use cases. Thus, the generalization of TSN is bought at the price of an enormous configuration space. In some use cases, this problem of large configuration space may be mitigated by plug-and-play features of IEEE 802.1 and the definition of application profiles (e.g., IEEE 801.1BA in AVB). In the case of use cases of TSN for CPS, the use of plug-and-play is much more limited (at least at the time of this writing) and most of the knobs and dials need to be explicitly configured.

Sometimes the standards hint at configuration possibilities but are not too elaborative in terms of configuration realization. However, as TSN is becoming a mainstream networking solution for CPS, it is reasonable to expect that over time more and more industrial and academic guidance will be published on how to configure certain aspects of TSN to achieve the desired effects.

10) Summary of the TSN General Review: We summarize our review of TSN in Table 3. We use a scoring system from one to five stars, one being the lowest score and five the maximum. This scoring is informal and only represents the perspective of the authors of this paper. The discussion of the items in the previous sections serves as an argumentation for the individual scores.

As depicted, we see the top strengths of TSN in the areas of communication bandwidth and scalability (based on the Ethernet heritage) closely followed by the real-time communication capabilities and the flexibility for reconfiguration. We interpret TSN's capabilities in terms of reliability and availability as medium. Finally, TSN has limitations in terms of integrity, confidentiality, and configuration ease. Consequently, in its current state, TSN cannot be directly deployed in safety-critical CPS without further design considerations (although TSN can be part of an overall solution for safety-critical CPS).

B. TSN Research Challenges

Following the discussion of strengths and weaknesses of TSN, several promising research directions with respect to TSN evolve. We discuss some of them in the following.

1) *Configuration Synthesis:* Probably, the most stringent research direction is configuration synthesis for TSN networks, in order to tackle TSN's shortcomings in configuration ease. On the one hand, this research direction addresses the question of how to use which TSN mechanism or combination of mechanisms to achieve which effect. On the other hand, this research direction targets to find novel solutions to scalability of the configuration, e.g., scalability of network schedules. Furthermore, the generation of TSN configurations with real-time and/or fault-tolerance requirements is another promising aspect of research in configuration synthesis.
2) *Dependability Improvements:* As indicated by the medium scores in integrity and availability, the TSN deployment in safety-critical systems is not straightforward. Significant improvements in those categories need to be achieved. Furthermore, safety-critical systems usually require the design of critical components of the system to follow design guidelines. Thus, the assurance of the design of TSN components needs to be studied as well.
3) *Security Improvements:* Although there is ongoing cooperation between the IEEE 802.1 TSN task group and the IEEE 802.1 security task group to investigate strategies to improve security capabilities of TSN, further research in TSN security is highly needed. TSN is an enabling technology to integrate IT (information technology) and OT (operations technology). Consequently, previously closed CPS will become more and more open and will be exposed to novel attacks. Continuous security research for TSN is essential to detect threats early on and to propose mitigation strategies.
4) *Verification and Validation:* As TSN is a rather new technology, with first products just entering various markets, it is essential that the technology and its products are assessed as much as possible from as many independent parties as possible. In particular, performance studies, simulation, prototypes, use case demonstrations, and so on, are desperately needed to grow the maturity of TSN solutions.

VII. CONCLUSION

In this paper, we reviewed and discussed the IEEE 802.1 TSN in industrial communication and automation systems. We provided an overview of the existing projects and then focused, with a critical perspective, on some core TSN standards that we consider highly relevant for industrial applications. We also briefly discussed application areas and elaborated on industrial automation and automotive applications in more detail. As TSN is a rather novel technology, academic research is just taking up speed in this direction. We provided insights on promising research activities in TSN and we expect that this paper will contribute to accelerating and fostering further research. ■

REFERENCES

[1] M. D. J. Teener *et al.*, "Heterogeneous networks for audio and video: Using IEEE 802.1 audio video bridging," *Proc. IEEE*, vol. 101, no. 11, pp. 2339–2354, Nov. 2013.

[2] A. Nasrallah *et al.*, "Ultra-low latency (ULL) networks: The IEEE TSN and IETF DetNet standards and related 5G ULL research," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 88–145, 1st Quart., 2018.

[3] X. Jiang *et al.*, "Low-latency networking: Where latency lurks and how to tame it," *Proc. IEEE*, vol. 107, no. 2, pp. 280–306, Feb. 2019.

[4] L. Seno, F. Tramarin, and S. Vitturi, "Performance of industrial communication systems: Real application contexts," *IEEE Ind. Electron. Mag.*, vol. 6, no. 2, pp. 27–37, Jun. 2012.

[5] L. Lo Bello, R. Mariani, S. Mubeen, and S. Saponara, "Recent advances and trends in

on-board embedded and networked automotive systems," *IEEE Trans. Ind. Informat.*, vol. 15, no. 2, pp. 1038–1051, Feb. 2019.

[6] J. Farkas, L. L. Bello, and C. Gunther, "Time-sensitive networking standards," *IEEE Commun. Standards Mag.*, vol. 2, no. 2, pp. 20–21, Jun. 2018.

[7] N. Finn, "Introduction to time-sensitive networking," *IEEE Commun. Standards Mag.*, vol. 2, no. 2, pp. 22–28, Jun. 2018.

[8] J. L. Messenger, "Time-sensitive networking: An introduction," *IEEE Commun. Standards Mag.*, vol. 2, no. 2, pp. 29–33, Jun. 2018.

[9] *IEEE Standard for Local and Metropolitan Area Network–Bridges and Bridged Networks*, IEEE Standard 802.1Q-2018 (Revision of IEEE Standard 802.1Q-2014), Jul. 2018, pp. 1–1993.

[10] S. Samii and H. Zinner, "Level 5 by Layer 2: Time-sensitive networking for autonomous vehicles," *IEEE Commun. Standards Mag.*, vol. 2, no. 2, pp. 62–68, Jun. 2018.

[11] J. Specht and S. Samii, "Urgency-based scheduler for time-sensitive switched ethernet networks," in *Proc. IEEE 28th Euromicro Conf. Real-Time Syst. (ECRTS)*, Jul. 2016, pp. 75–85.

[12] G. M. Garner and H. Ryu, "Synchronization of audio/video bridging networks using IEEE 802.1AS," *IEEE Commun. Mag.*, vol. 49, no. 2, pp. 140–147, Feb. 2011.

[13] K. B. Stanton, "Distributing deterministic, accurate time for tightly coordinated network and software applications: IEEE 802.1AS, the TSN profile of PTP," *IEEE Commun. Standards Mag.*, vol. 2, no. 2, pp. 34–40, Jun. 2018.

[14] M. Gutiérrez, W. Steiner, R. Dobrin, and S. Punnekkat, "Synchronization quality of IEEE 802.1AS in large-scale industrial automation networks," in *Proc. IEEE Real-Time Embedded Technol. Appl. Symp. (RTAS)*, Apr. 2017, pp. 273–282.

[15] G. Gaderer, A. Treytl, and T. Sauter, "Security aspects for IEEE 1588 based clock synchronization protocols," in *Proc. IEEE Int. Workshop Factory Commun. Syst. (WFCS)*, Torino, Italy, Jun. 2006, pp. 247–250.

[16] A. Treytl and B. Hirschler, "Security flaws and workarounds for IEEE 1588 (transparent) clocks," in *Proc. Int. Symp. Precision Clock Synchronization Meas., Control Commun. (ISPCS)*, Oct. 2009, pp. 1–6.

[17] T. Mizrahi, *Security Requirements of Time Protocols in Packet Switched Networks*, RFC 7384, Internet Requests for Comments, RFC Editor, Oct. 2014. [Online]. Available: http://www.rfc-editor.org/rfc/rfc7384.txt

[18] E. Lisova *et al.*, "Protecting clock synchronization: Adversary detection through network monitoring," *J. Elect. Comput. Eng.*, vol. 2016, Mar. 2016, Art. no. 6297476.

[19] L. Lamport and P. M. Melliar-Smith, "Byzantine clock synchronization," in *Proc. 3rd Annu. ACM Symp. Princ. Distrib. Comput.*, 1984, pp. 68–74.

[20] W. Steiner and B. Dutertre, "The TTEthernet synchronisation protocols and their formal verification," *Int. J. Crit. Comput.-Based Syst.*, vol. 4, no. 3, pp. 280–300, 2013.

[21] G. Prytz, "A performance analysis of EtherCAT and PROFINET IRT," in *Proc. IEEE Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, Sep. 2008, pp. 408–415.

[22] H. Kopetz, A. Ademaj, P. Grillinger, and K. Steinhammer, "The time-triggered ethernet (TTE) design," in *Proc. 8th IEEE Int. Symp. Object-Oriented Real-Time Distrib. Comput. (ISORC)*, May 2005, pp. 22–33.

[23] P. Pedreiras, P. Gai, L. Almeida, and G. C. Buttazzo, "FTT-Ethernet: A flexible real-time communication protocol that supports dynamic QoS management on Ethernet-based systems," *IEEE Trans Ind. Informat.*, vol. 1, no. 3, pp. 162–172, Aug. 2005.

[24] H. Kopetz and G. Bauer, "The time-triggered architecture," *Proc. IEEE*, vol. 91, no. 1, pp. 112–126, Jan. 2003.

[25] F. Consortium *et al.*, "FlexRay communications system-protocol specification," *Version*, vol. 2, no. 1, pp. 198–207, 2005.

[26] D. Tămaş-Selicean, P. Pop, and W. Steiner, "Design optimization of TTEthernet-based distributed real-time systems," *Real-Time Syst.*, vol. 51, no. 1, pp. 1–35, 2015.

[27] Z. Hanzálek, P. Burget, and P. Sucha, "Profinet IO IRT message scheduling with temporal constraints," *IEEE Trans Ind. Informat.*, vol. 6, no. 3, pp. 369–380, Aug. 2010.

[28] B. Tanasa, U. D. Bordoloi, P. Eles, and Z. Peng, "Scheduling for fault-tolerant communication on the static segment of FlexRay," in *Proc. IEEE 31st Real-Time Syst. Symp. (RTSS)*, Nov./Dec. 2010, pp. 385–394.

[29] H. Zeng, M. Di Natale, A. Ghosal, and A. Sangiovanni-Vincentelli, "Schedule optimization of time-triggered systems communicating over the FlexRay static segment," *IEEE Trans. Ind. Informat.*, vol. 7, no. 1, pp. 1–17, Feb. 2011.

[30] W. Steiner, "An evaluation of SMT-based schedule synthesis for time-triggered multi-hop networks," in *Proc. 31st IEEE Real-Time Syst. Symp.*, Dec. 2010, pp. 375–384.

[31] S. S. Craciunas, R. S. Oliver, M. Chmelík, and W. Steiner, "Scheduling real-time communication in IEEE 802.1Qbv time sensitive networks," in *Proc. ACM 24th Int. Conf. Real-Time Netw. Syst.*, 2016, pp. 183–192.

[32] S. S. Craciunas, R. S. Oliver, and W. Steiner. (2017). "Formal scheduling constraints for time-sensitive networks." [Online]. Available: https://arxiv.org/abs/1712.02246

[33] R. S. Oliver, S. S. Craciunas, and W. Steiner, "IEEE 802.1Qbv gate control list synthesis using array theory encoding," in *Proc. IEEE Real-Time Embedded Technol. Appl. Symp. (RTAS)*, Apr. 2018, pp. 13–24.

[34] W. Steiner, S. S. Craciunas, and R. S. Oliver, "Traffic planning for time-sensitive communication," *IEEE Commun. Standards Mag.*, vol. 2, no. 2, pp. 42–47, Jun. 2018.

[35] *IEEE Standard for Local and metropolitan area networks—Time-Sensitive Networking for Fronthaul*, IEEE Standard 802.1CM-2018, Jun. 2018, pp. 1–62.

[36] *IEEE Standard for Local and Metropolitan Area Networks—Station and Media Access Control Connectivity Discovery*, IEEE Standard 802.1AB-2016 (Revision of IEEE Std 802.1AB-2009), Mar. 2016, pp. 1–146.

[37] T. Wan and P. Ashwood-Smith, "A Performance Study of CPRI over Ethernet with IEEE 802.1Qbu and 802.1Qbv Enhancements," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2015, pp. 1–6.

[38] D. Thiele and R. Ernst, "Formal worst-case performance analysis of time-sensitive ethernet with frame preemption," in *Proc. IEEE 21st Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, Sep. 2016, pp. 1–9.

[39] *IEEE Standard for Local and Metropolitan Area Networks—Audio Video Bridging (AVB) Systems*, IEEE Standard 802.1BA-2011, Sep. 2011, pp. 1–45.

[40] *Industrial Communication Networks—High Availability Automation Networks—Part 3: Parallel Redundancy Protocol (PRP) and High-Availability Seamless Redundancy (HSR)*, Standard IEC 62439-3:2016, 2016.

[41] S. Bryant and P. Pate, *Pseudo Wire Emulation Edge-to-Edge (PWE3) Architecture*, document RFC 3985, Internet Requests for Comments, RFC Editor, Mar. 2005.

[42] IEEE 802.1 Plenary, Geneva, Switzerland. (Jul. 2013). Formal Analysis of P802.1CB. Accessed: Jan. 18, 2019. [Online]. Available: http://www.ieee802.org/1/files/public/docs2013/new-avb-wsteiner-8021CB-formal-analysis-0713-v01.pdf

[43] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Trans. Depend. Sec. Comput.*, vol. 1, no. 1, pp. 11–33, Jan./Mar. 2004.

[44] J. H. Saltzer, D. P. Reed, and D. D. Clark, "End-to-end arguments in system design," *ACM Trans. Comput. Syst.*, vol. 2, no. 4, pp. 277–288, 1984.

[45] *Proposal for P802.1Qcc Control Flows*. Accessed: Aug. 24, 2018. [Online]. Available: http://www.ieee802.org/1/files/public/docs2014/cc-nfinn-control-flows-0414-v02.pdf

[46] T. Hannelius, M. Salmenpera, and S. Kuikka, "Roadmap to adopting OPC UA," in *Proc. 6th IEEE Int. Conf. Ind. Inform. (INDIN)*, Jul. 2008, pp. 756–761.

[47] M. Bjorklund, *The YANG 1.1 Data Modeling Language*, document RFC 7950, 2016.

[48] M. Gutiérrez, A. Ademaj, W. Steiner, R. Dobrin, and S. Punnekkat, "Self-configuration of IEEE 802.1 TSN networks," in *Proc. 22nd IEEE Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, Sep. 2017, pp. 1–8.

[49] N. McKeown *et al.*, "OpenFlow: Enabling innovation in campus networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, Apr. 2008.

[50] J. Farkas, S. Haddock, and P. Saltsidis. (2014). "Software defined networking supported by IEEE 802.1Q." [Online]. Available: https://arxiv.org/abs/1405.6953

[51] M. Wollschlaeger, T. Sauter, and J. Jasperneite, "The future of industrial communication: Automation networks in the era of the Internet of Things and industry 4.0," *IEEE Ind. Electron. Mag.*, vol. 11, no. 1, pp. 17–27, Mar. 2017.

[52] E. Schweissguth, P. Danielis, C. Niemann, and D. Timmermann, "Application-aware industrial Ethernet based on an SDN-supported TDMA approach," in *Proc. IEEE World Conf. Factory Commun. Syst. (WFCS)*, May 2016, pp. 1–8.

[53] S. Schriegel, T. Kobzan, and J. Jasperneite, "Investigation on a distributed SDN control plane architecture for heterogeneous time sensitive networks," in *Proc. 14th IEEE Int. Workshop Factory Commun. Syst. (WFCS)*, Jun. 2018, pp. 1–10.

[54] C. Simon, M. Maliosz, and M. Mate, "Design aspects of low-latency services with time-sensitive networking," *IEEE Commun. Standards Mag.*, vol. 2, no. 2, pp. 48–54, Jun. 2018.

[55] *CPRI Specification*. Accessed: Aug. 30, 2018. [Online]. Available: http://www.cpri.info/spec.html

[56] A. Neumann, L. Wisniewski, R. S. Ganesan, P. Rost, and J. Jasperneite, "Towards integration of industrial ethernet with 5G mobile networks," in *Proc. 14th IEEE Int. Workshop Factory Commun. Syst. (WFCS)*, Jun. 2018, pp. 1–4.

[57] M. Jakovljevic, A. Geven, N. Simanic-John, and D. M. Saatci, "Next-gen train control/management (TCMS) architectures: Drive-by-data system integration approach," in *Proc. 9th Eur. Congr. Embedded Real-Time Softw. Syst.*, 2018.

[58] *Time-Triggered Ethernet Standard AS6802*. Accessed: Aug. 30, 2018. [Online]. Available: https://www.sae.org/standards/content/as6802

[59] *Avionics Full Duplex Ethernet and the Time Sensitive Networking Standard*. Accessed: Aug. 30, 2018. [Online]. Available: http://www.ieee802.org/1/files/public/docs2015/TSN-Schneele-AFDX-0515-v01.pdf

[60] W. Steiner, P. Heise, and S. Schneele, "Recent IEEE 802 developments and their relevance for the avionics industry," in *Proc. IEEE/AIAA 33rd Digit. Avionics Syst. Conf. (DASC)*, Oct. 2014, pp. 2A2-1–2A2-12.

[61] *Requirements IEC/IEEE 60802*. Accessed: Jan. 18, 2019. [Online]. Available: http://www.ieee802.org/1/files/public/docs2018/60802-industrial-requirements-1218-v12.pdf

[62] *Use Cases IEC/IEEE 60802*. Accessed: Jan. 18, 2019. [Online]. Available: http://www.ieee802.org/1/files/public/docs2018/60802-industrial-use-cases-0918-v13.pdf

[63] P. Pop, M. L. Raagaard, M. Gutierrez, and W. Steiner, "Enabling fog computing for industrial automation through time-sensitive networking (TSN)," *IEEE Commun. Standards Mag.*, vol. 2, no. 2, pp. 55–61, Jun. 2018.

[64] G. Alderisi, A. Caltabiano, G. Vasta, G. Iannizzotto, T. Steinbach, and L. Lo Bello, "Simulative assessments of IEEE 802.1 Ethernet AVB and time-triggered ethernet for advanced driver assistance systems and in-car infotainment," in

Proc. Veh. Netw. Conf., Nov. 2012, pp. 187–194.

[65] G. Alderisi, G. Iannizzotto, and L. Lo Bello, "Towards IEEE 802.1 Ethernet AVB for Advanced Driver Assistance Systems: A preliminary assessment," in *Proc. IEEE 17th Conf. Emerg. Technol. Factory Autom.*, Sep. 2012, pp. 1–4.

[66] L. Lo Bello, "Novel trends in automotive networks: A perspective on Ethernet and the IEEE Audio Video Bridging," in *Proc. 19th IEEE Int. Conf. Emerg. Technol. Factory Autom.*, Sep. 2014, pp. 1–8.

[67] G. Alderisi, G. Patti, and L. Lo Bello, "Introducing support for scheduled traffic over IEEE audio video bridging networks," in *Proc. 18th IEEE Conf. Emerg. Technol. Factory Autom.*, Sep. 2013, pp. 1–9.

[68] M. Ashjaei, G. Patti, M. Behnam, T. Nolte, G. Alderisi, and L. Lo Bello, "Schedulability analysis of Ethernet audio video bridging networks with scheduled traffic support," *Real-Time Syst.*, vol. 53, no. 4, pp. 526–577, Jul. 2017.

[69] *RazorMotion Product Flyer*. Accessed: Aug. 24, 2018. [Online]. Available: https://www.tttech-auto.com/wp-content/uploads/TTTech-Automotive_RazorMotion-1.pdf

ABOUT THE AUTHORS

Lucia Lo Bello received the M.S. degree in electronic engineering and the Ph.D. degree in computer engineering from the University of Catania, Catania, Italy, in 1994 and 1998, respectively.

In 2014, she was a Guest Professor with the University of Malardalen, Västerås, Sweden. From 2000 to 2001, she was a Visiting Researcher with the Department of Computer Engineering, Seoul National University, Seoul, South Korea. Since 2004, she has been actively involved in standardization activities, relevant to wired and wireless industrial networks, at both national and international levels. She is currently a tenured Associate Professor with the Department of Electrical, Electronic and Computer Engineering, University of Catania. Her current research interests include automotive communications, with a special focus on automotive Ethernet, IEEE Audio Video Bridging and Time-Sensitive Networking, industrial networks, real-time embedded systems, and wireless sensor networks. She has authored or coauthored more than 150 technical papers in these areas. She is responsible for several international and national projects in the area of real-time embedded systems and networks.

Dr. Lo Bello is a Member-at-Large of the Industrial Electronics Society (IES) AdCom. She was the Chair of the IES Technical Committee on Factory Automation for two terms, from 2014 to 2015 and from 2016 to 2017. She is the current IES representative within IEEE Women in Engineering.

Wilfried Steiner received the Doctor of Technical Sciences degree and the Venia Docendi degree in computer science from the Vienna University of Technology, Vienna, Austria, in 2005 and 2018, respectively.

He is currently a Corporate Scientist with TTTech Computertechnik AG, Vienna, and the Leader of the TTTech Labs, Vienna. His research is focused on dependable cyber-physical systems (CPS) and he significantly contributed to the domains of automotive, space, aerospace, as well as new energy and industrial automation. He designs algorithms and protocols with real-time, dependability, and security requirements. In particular, he follows a model-driven design approach in which he applies formal methods, such as model-checking, SMT-solving, and theorem proving to obtain formal correctness proofs of the solutions developed. While the initial targets of his research have been rather traditional dependable CPS, such as automobiles and airplanes, more recently he also addresses research problems in the area of Internet of Things (IoT), like the Industrial IoT and Industrie 4.0, as well as autonomous vehicles. He has been a Voting Member in the IEEE 802.1 Working Group from 2013 to 2016.