Constructive or Optimized: An Overview of Strategies to Design Networks for Time-Critical Applications

Gavriluţ, Voica; Pruski, Aleksander; Berger, Michael Stübert

Published in:
ACM Computing Surveys

Link to article, DOI:
10.1145/3501294

Publication date:
2023

Document Version
Publisher's PDF, also known as Version of record

Constructive or Optimized: An Overview of Strategies to Design Networks for Time-Critical Applications

VOICA GAVRILUŢ and ALEKSANDER PRUSKI, Comcores Aps. and Technical University of Denmark (DTU)
MICHAEL STÜBERT BERGER, Technical University of Denmark (DTU)

Distributed systems are pervasive nowadays, being found in different areas, from smart toys to smart factories. As the usage of such systems increases, so does their complexity and design. Therefore, this work aims to overview the methods for designing networks that accommodate time-constrained distributed applications.

The work starts with a history of time-aware Ethernet-based protocols. Then, it continues with an overview of the design strategies from the literature. For each research paper, there are presented the model, addressed problem, exploration strategy, and results. Furthermore, for each type of problem are identified the constructive and optimization design strategies. Regarding the results, this work investigates the improvements of reliability, timeliness, and network cost.

CCS Concepts: • **Computing methodologies**; • **Networks** → *Network performance evaluation*; *Network protocols*;

Additional Key Words and Phrases: Time-sensitive networking, network performance improvement, network design strategies, heuristic methods, metaheuristic methods

ACM Reference format:
Voica Gavriluţ, Aleksander Pruski, and Michael Stübert Berger. 2022. Constructive or Optimized: An Overview of Strategies to Design Networks for Time-Critical Applications. *ACM Comput. Surv.* 55, 3, Article 62 (February 2022), 35 pages.
https://doi.org/10.1145/3501294

1 INTRODUCTION

Distributed computing systems are pervading in our society, with units being found e.g., in smart toys, home appliances, city infrastructure, medical devices, vehicles, industrial sensors, and actuators. Those units, with each essentially being a computer, interact with each other to form a networked system aiming to fulfil a shared set of goals. As the usage of distributed systems expands, they become increasingly complex. Following this trend, the methods for designing such systems are also constantly evolving. Additionally, for some of those systems, the requirements

This work is partially supported by Comcores ApS and Innovationsfonden Denmark through grants 8053-00031B and 9066-00048B.
Authors' addresses: V. Gavriluţ and A. Pruski, Comcores Aps., Haldor Topsøes Allé 1, Kgs. Lyngby, 2800 Denmark and Technical University of Denmark (DTU), Ørsteds Plads 343, Kgs. Lyngby, 2800 Denmark; emails: voga@dtu.dk, alper@fotonik.dtu.dk; M. S. Berger, Technical University of Denmark (DTU), Ørsteds Plads 343, Kgs. Lyngby, 2800 Denmark; email: msbe@fotonik.dtu.dk.

0360-0300/2022/02-ART62 $15.00
https://doi.org/10.1145/3501294

placed on the inter-networking are remarkably strict. Therefore, the goal of this work is to provide an overview of existing methods for designing distributed time-critical systems.

There are contexts [25] where: (1) human beings and machineries have to co-exist, (2) digital systems are used to enhance well-being of people, or (3) transportation relies on smart vehicles. For the first use-case, since human behavior is rather unpredictable, the machines have to be designed to adapt to human inputs. Even if this scenario seems futuristic, it already started to be used today in smart cities and smart factories [58]. The second use-case encompasses systems that generally improve quality-of-life and safety [24, 77]. Common examples would be remote nannies and other monitoring equipment, especially for patients in a hospital. The majority of those use cases would be considered having only high **Quality-of-Service** (**QoS**) requirements.

Vehicles with **Advanced Driver Assistance Systems** (**ADASes**), airplanes with flight-by-wire systems and self-driven trains all fall into the third use-case [88]. In addition to the safety of the passengers, the smart terrestrial vehicles must also take into account safety of pedestrians and other road-users. To that end, an international standard ISO 26262 [52] describes the functional safety for electrical, electronic, and software sub-systems used in manufacturing of road vehicles. ISO 26262 classifies the safety risks and malfunction countermeasures into five **Automotive Safety Integrity Levels** (**ASIL**). Level 0 covers quality management with no safety requirements, whereas on the other side of the spectrum (level 4) ASIL D groups elements with highest safety requirements. Those are sub-systems whose failure severely threatens human-life or leads to fatal injuries.

Usually, systems that were designed considering timing constraints, are called real-time systems. But there is also one common misconception, that real-time systems have to react fast [11]. A more correct understanding, is that real-time systems must react within appropriate time. Therefore, this work addresses the design techniques of distributed systems that have to behave predictable—which means that timing requirements of the applications distributed over the network are satisfied even in the worst-case scenarios.

Alongside latency, bandwidth demands are also constantly increasing. For example, both autonomous driving and driver assistance systems rely on data from multitude of sensors, such as cameras, lidars, and radars. Those require link-rates of up to tens Gbps [78]. Similar trend is observed in fronthaul of mobile networks, where envisioned requirements reach hundreds Gbps in order to support 5G technologies [1].

Due to the substantial growth in complexity, and the need to reduce costs, such timing-deterministic applications have to be deployed over mixed-criticality systems. In a mixed-criticality system, the applications of different timing requirements share the same infrastructure. Traditional (safe and expensive) approach was to completely isolate components and infrastructures with different goals and requirements. In an airplane, for example, for the flight-by-wire system, each functionality was implemented on its own computational unit and those units were connected point-to-point. This approach, besides being very expensive, increases the number of computers and wiring adding thus to the aircraft weight. Thus, with a shared infrastructure, the wiring and the number of networking components are reduced. This shared network is then "sliced", meaning that each subsystem sees it as exclusive. Each slice must provide enough capacity and fulfil timing requirements particular to the sub-system that it serves.

Another trend is to move from mass production towards mass customization. This trend can be observed even for areas with timing-deterministic requirements where are used mixed-criticality networking components based on adapted **Commercial Off-The-Shelf** (**COTS**) components. The increases in demands and complexity of distributed systems make harder and harder the job of designing such networks. The inherently increasing complexity of distributed systems leads to an increase on the non-recurring engineering costs of networking components. However, those costs can be offset by avoiding vendor lock-in and reducing the operational costs.

Besides having an infrastructure that is aware of timeliness and reliability, some configurations should be synchronized across the network. This coordination of configuration parameters for the network components—or network design as we are calling it through this article —is part of a wider process known as network orchestration. There is an entity, called **Network Controller** (**NC**), which orchestrates a network configuration by maintaining a global view of the network (through network monitoring and discovery) deciding the optimal configuration parameters for each component (through network design) and conveying these configuration parameters to each network node (through network management). The network design comes under the responsibility of **Control and Management** (**C&M**) planes. An example description of C&M planes can be found in RFC7426 [26] that specifies the **Software-Defined Networking** (**SDN**) architecture. According to this specification, the control plane "instructs network devices with respect to how to process and forward packets"; while the management plane "monitors, configures, and maintains" ports of the network node.

Ethernet [36] is a widely used communication protocol characterized by low costs and high speeds. It cannot, however, satisfy the requirements of **timing-critical** (**TC**) applications [15]. To overcome these limitations, multiple extensions were proposed. Some of them are as **Avionics Full-DupleX** (**AFDX**) switched Ethernet [4], **Time-Triggered Ethernet** (**TTEthernet**) [73], **Audio-Video Bridging** (**AVB**) [37], and **Time-Sensitive Networking** (**TSN**). They all aim for increasing the reliability and timing predictability of data transmission. Our main focus is on TSN, which is a set of IEEE standards and amendments that extend Ethernet for dependable distributed applications. For completeness and better understanding of TSN and the current design methodologies, we will first explore TSN's ancestors such as switched Ethernet, AFDX, TTEthernet, and AVB.

We are focusing on TSN since it is announced to be a good candidate as underlying technology for Internet of the future [2]. TSN is introduced to address timing and reliability requirements for all previously mentioned time-sensitive areas. This means that features (defined as objects, data manipulation procedures, and transmission mechanisms) specified in different amendments must be combined to address the requirements of a certain area. Therefore, TSN profiles define default values and feature combinations for particular areas. Such profiles are IEEE 802.1CM [44] (for fronthaul networks), IeC/IEEE 60802 [46] (for industrial automation), IEEE 802.1DG [48] (for automotive in-vehicle networks), and IEEE 802.1DP [49] (for avionics onboard networks).

Since the TSN set of standards is relatively young, and other Ethernet extensions evolved in different environments, mature strategies for designing networks that can accommodate TC applications are sparse. As system requirements and protocols are evolving, so do the need for methodologies of networks design. Currently, the traffic demands are increasing, i.e., more throughput is required, reliability is upper bounded (tolerating for example at most one failure in 10^9 hours of operation [84]) or nearly jitter-free end-to-end transmissions. For going further on design methodologies, we need first to establish a "terra firma" of the strategies used so-far. Although, design strategies exist also for switched Ethernet, AFDX, and TTEthernet, throughout this methodology investigation, we are focusing on strategies proposed for TSN-based networks.

In order to conduct this methodology overview, the state-of-the-art methods are investigated, and for each the advantages and limitations are described. Please note that a design strategy does not refer only to the algorithm. Therefore, the research papers are analyzed in terms of data structures (or model), addressed design problem, exploration strategy, and obtained results. In terms of network and traffic models, the satisfied constraints, proposed design methods and obtained results are presented. Furthermore, in terms of results, we are studying the improvements of (1) the monetary cost of the network, (2) traffic throughput, (3) **Worst-Case end-to-end Delay** (**WCD**), and (4) the end-to-end jitter.

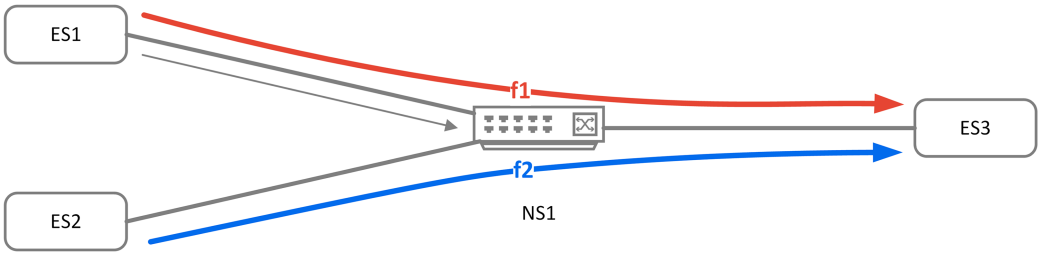

Fig. 1. Network model that represents a network with 3 ESes and 1 NS. The Switch has 3 Physical Ports; each Being Directly Connected with an ES. A DL is Represented with a Thin Gray Arrow and a DP is represented with a Thin Red Arrow.

This overview article has the following structure. Section 2 defines the architecture and traffic models. The Ethernet-based communication protocols that lead to TSN are presented in Section 3. The design problems are formulated in Section 4. The key contribution of our article is in Section 5, which presents the most relevant strategies used to design time-dependable networks, and in Section 6, which discusses the achievements of the analyzed strategies and proposes future research directions. Finally, Section 7 concludes the article.

2 SYSTEM MODEL

In this section, we are introducing the formal description of the network. Based on that network model, we will present details about the traffic model. Particularly, this explanation will focus on traffic properties, given by applications, and requirements, dictated by the distributed system or usage area.

2.1 Network Model

Since this work focuses mainly on extensions of switched Ethernet, we start with introducing the general model of an Ethernet-based network. A network is mathematically represented as an undirected graph $G(\mathcal{N}, \mathcal{E})$ of network nodes $\mathcal{N}$ that are interconnected by edges $\mathcal{E}$. Because in the community there is no consensus regarding the notations, we generically denote a network node nn which represents an **End System** (**ES**) or **Network Switch** (**NS**), the sets of ESes and NSes being denoted $\mathcal{ES}$ and $\mathcal{NS}$, respectively. The edges represent the full-duplex connections between two network nodes. Figure 1 illustrates a network composed of 3 ESes and 1 NS.

The ES represents a sensor, an activator, or a computation unit. An ES usually comprises a processor, possibly multi-core; a memory; and I/Os. Mainly only the computation units are in charge of intelligence, though, currently, smart sensors are also used. These sensors filter out the noise by pre-processing the passed-through data.

A NS (bridge in IEEE nomenclature) is responsible for connecting (bridging) network segments together. We take a modern view of an NS where the network segment is a single full-duplex link. A more legacy view is provided in Section 3. Typical Ethernet switch will have multiple ports (three in Figure 1), and is responsible at least for:

— learning the address of attached network nodes,
— forwarding traffic to destinations in accordance to learned information.

A detailed logical decomposition of an NS is available in the Ethernet standard [34, Section 8].

For the purpose of this article, we take a simplistic view, presented in Figure 2. Here, each port consists of an ingress port and an egress port. The ingress port is responsible for packet

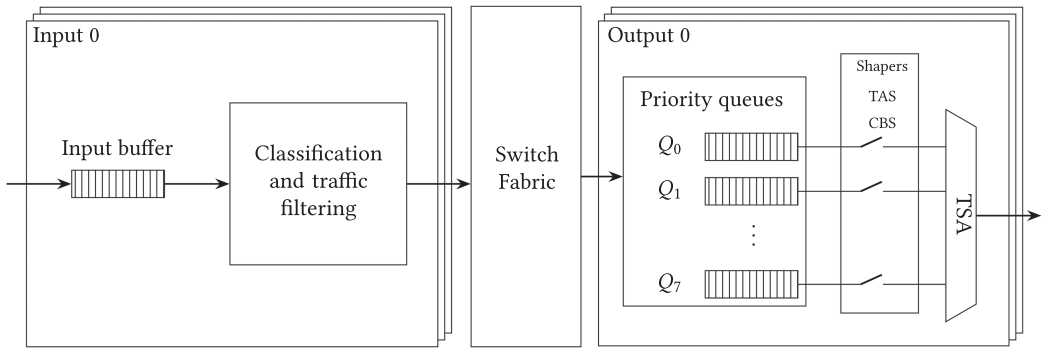

Fig. 2. Structure of a switch where a physical port is logically divided into ingress and egress ports. Ingress is Responsible for Traffic Classification and Policing and Egress is Responsible for Traffic Shaping.

classification and policing of misbehaving flows. Packets are then switched to the egress ports, where they are scheduled for transmission according to their class and chosen set of policies.

The presented functional model of the NS abstracts away the complexities of actual physical implementation. Most of the work regarding TSN does not consider buffering structures necessary to support line-rate switching operations and instead assumes Output Queueing model (even if emulated). Output Queueing in this case means, that the switch fabric between ingress and egress ports guarantees enough performance for the delay resulting from switching to be negligible [75, Chapter 17]. Furthermore, the output Queueing emulation can be achieved with shared-memory architectures. However, with increasing port-densities and rates, the memory becomes the limiting factor of this queuing model. Nevertheless, for the purpose of this work, we further consider the output queuing model and assume that the delay stemming from switch-fabric contention is negligible.

As stated before, a connection is represented by a full-duplex link that allows simultaneous transmission of data in both directions. To capture the direction of data though, we will use the concept of **Dataflow Link** (**DL**)—where dl_{nn_i,nn_j} means that node nn_i sends data to nn_j. In Figure 1, dl_{es_1,ns_1} is represented by a thin arrow in gray.

Furthermore, to show the flow of data between two ESes, we introduce the notion of **Dataflow Path** (**DP**). A DP is a sequence of DLs, the first DL starting from an ES and the last DL ending in another ES. For example, the DP $[dl_{es_1,ns_1}, dl_{ns_1,es_3}]$ is represented in 1 with a thin red arrow. The set of DLs and DPs is denoted with $\mathcal{DL}$ and $\mathcal{DP}$, respectively.

There are multiple types of links and switches available on the market, and the collection of those components available for building the network is denoted as **Components Library** (**CL**). Link type represents its speed and cost. Similarly, a switch type is characterized by number of ports, their supported rates, and a set of available traffic shapers, and other QoS tools. In Section 3, we provide more details about shapers and monetary cost.

2.2 Traffic Model

Through this article, the traffic model captures the flows of data that are exchanged among applications. TC flows denote the flows produced or consumed by at least one TC application.

In our model, a flow f_i has the following tuple of properties $(src_i, dests_i, P_i, T_i)$. The flow f_i is sent by one source src_i ES and received by possible multiple destination ESes. The data is transmitted from source to destinations through a route, r_i denoting the route of f_i. If the flow is consumed by a single destination, we call it a unicast transmission. In this case, the route consists

Table 1. Example Traffic Properties; Each Flow Characterized by Source, Destination, Flow Size, Number of Frames Per Instance, and Period

Id	*src*	*dests*	P (in B)	k	T (in μs)
f_1	es_1	$\{es_3\}$	1,500	1	250
f_2	es_2	$\{es_3\}$	2,000	2	250

of a single DP. In contrast, when data is disbursed at multiple destinations, the transmission is multi-cast, and the route is a multi-cast tree. A tree represents a set of DPs starting from the same ES and reaching all destinations.

Furthermore, the size represents the maximum number of bytes that a flow can carry on. On an Ethernet-based network, flows are packed and transmitted in Ethernet frames. If the size of a flow exceeds the Ethernet-dictated **Maximum Transmission Unit** (**MTU**), the flow is split into multiple frames, k_i denoting the number of frames of a flow. $f_{i,j}|j \leq k_i$ represents the jth frame of flow f_i.

Usually, in a distributed system, the data is produced continuously, a piece of information produced at a time being called flow instance. The repetition is either strictly periodic (successive flow instances of same size separated by a fixed time period) or sporadic (successive flow instances of variating sizes separated by at least a minimum amount of time, known as **Minimum Inter-Arrival Time** (**MIAT**)). Please note that all frames of a flow inherit its period.

For example, the network in Figure 1 accommodates the traffic presented in Table 1. In this example, there are two flows f_1 and f_2, that are produced by the ESes es_1 and es_2, respectively. Both flows are unicast, being both received by es_3. They have different payload sizes of $1500B$ and $2000B$, respectively. For example, the payload of flow f_2 exceeds the MTU of $1500B$, a flow instance consisting thus of 2 frames. Furthermore, both flows have a similar period of 250 μs.

In a distributed system that have to support deterministic applications, the traffic has to satisfy some reliability and timeliness requirements. In order to assess if these requirements are met, there are introduced the requirements metrics such as **packet Loss Rate** (**PLR**)—for reliability—and jitter and delay—for timeliness.

The PLR is measured over a time interval and it represents the ratio between the number of lost packets and the total number of sent packets. The delay of a flow represents the time that it takes to reach all destinations relative to the time when the flow was forwarded from the source. Worth noticing is that, in the community, the upper bound of end-to-end delay is called deadline. The variation over time of the delay represents the flow jitter. For a flow f_i, the upper bounds on PLR, delay, and jitter are denoted PLR_i, D_i and J_i, respectively. Please note that if a flow is split in multiple frames, we consider that the deadline and jitter requirements are met if they are satisfied for each individual frame.

Table 2 illustrates the set of requirements for the traffic described in Table 1. As we can observe, both flows have a PLR of 10^{-7}, which means that at most 1 frame is acceptable to be lost out of ten million (10^7) frames transmitted. As highlighted in 2, it is not mandatory to have similar timeliness requirements for flows with same period. Namely, f_1 and f_2 have a deadline of 100 μs and 200 μs and a jitter of 70 μs and 100 μs, respectively.

3 THE ROAD TOWARDS TIMING-AWARE COMMUNICATION

This section presents protocols that are able to support distributed applications with timing and reliability constraints. In order to achieve such deterministic communication, they evolved

Table 2. Example Traffic Requirements; Each Flow Transmission Upper Bounded by Package Loss Rate, Deadline, and Jitter

Id	*PLR*	*D* (in μs)	*J* (in μs)
f_1	10^{-7}	100	70
f_2	10^{-7}	200	100

dramatically since the 70s. The history of machine-to-machine communication starts earlier though—in the 1960s. Even at the beginning of 2000s, most of the protocols were based on bus architectures. In those, multiple ESes share a common transmission medium. The protocols, that focused on reliability and timeliness (in chronological order), are: **Controller Area Network** (**CAN**) [71], Fieldbus [51], SAFEbus [28], **Time-Triggered Protocol** (**TTP**) [56], **Ethernet for Control Automation Technology** (**EtherCAT**) [18], **Flexible Time-Triggered Ethernet** (**FTT-Ethernet**), [66] and FlexRay [12].

Starting from mid 90s, the protocols that can support deterministic communication, based on *switched architectures* have emerged. Examples are: ProfiNet [50], AFDX switched Ethernet [4], AVB [37], TTEthernet [73], and TSN [38].

This work focuses on modern communication protocols over switched architectures. The protocols listed above all extend Ethernet, trying to benefit from its high speed and low cost. One of the most crucial drawbacks of Ethernet is its unsuitability for applications with reliability and timeliness requirements [15]. Consequently, in the following sections we showcase the Ethernet extensions of AFDX, AVB, TTEthernet, and TSN, that focuses on overcoming this limitation. In order to aid understanding of those extensions, we start with presenting legacy shared-media and switched Ethernet.

In the following sub-sections, each protocol is presented highlighting the relevant architecture particularities, framing structure and transmission strategy. Please note that in order to preserve the evolutive presentation of the protocols, we are citing the specification or academic work that introduced for the first time the inherent characteristics of a protocol.

One of the aspects regarding timeliness is the so-called end-to-end transmission delay. The end-to-end delay is the sum of per-hop delays experienced along the transmission route. Furthermore, the per-hop delay consists of processing delay, queuing delay, serialization delay, and propagation delay. Among those, only queuing delay is directly affected by a transient network load, and therefore directly controllable with QoS techniques. With this being the case, it receives a special spotlight in the following protocols' presentations.

3.1 Traditional Ethernet

Ethernet is a communication protocol developed initially by Metcalfe and his team from Xerox Parc in early 70s. The protocol was patented in 1976 in the United States [60], and the first standard appeared a few years later, in 1985 [15, 83].

Its goal was to develop medium, network, and transmission procedures, to transmit the data originating on one station to all others attached to the network. The name Ethernet comes from *luminiferous ether* —a medium carrying the electromagnetic waves [53].

The initial version of Ethernet uses a shared medium; which means that *all* interacting stations are connected and have equal access to it at any instance of time. For describing this universal access to the communication media, there is used also the syntagm that all stations are on the same broadcast (here also equivalent to collision) domain. Originally, shared media Ethernet used coaxial

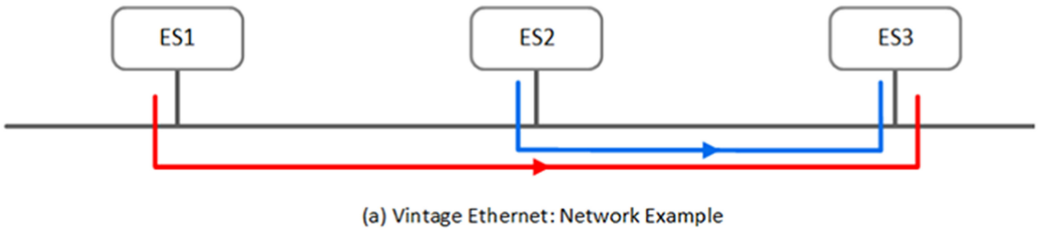

Fig. 3. Example of a shared media ethernet where 3 end stations are attached to a bus.

cables, supporting data rates up to 10 **Megabits per second** (**Mbps**). The topologies available were: bus, ring, and - those facilitated with hubs - the star and tree [83]. Hubs are effectively electrical signal splitters and repeaters, allowing multiple end stations connected to a single multi-port hub to see each other as being connected to a single bus.

Figure 3 shows an example of a bus architecture where 3 stations are inter-connected. Those stations use network adapters in order to *tap into* the bus. Later on, coaxial cables were replaced by Twisted Pair cables due to latter's lower cost.

Ethernet is a prominent example of a packet switched network—the data to be sent is split and packaged in Ethernet frames (those data chunks are called payloads). Each frame is prefixed with Ethernet header, and suffixed by trail, in a process known as encapsulation. The exact format of an Ethernet frame is specified in IEEE 802.3 [36], and summarized here.

A slice of information serving a particular function in either header or trail is called a field. The fields in the Ethernet header contain information about the intended receiver (destination address, 6 Bytes), sender (source address, 6 Bytes), and also specify either the length or type of payload (4 Bytes). The trail contains a Frame Check Sequence (4 Bytes) used to verify integrity of the frame upon reception. It is calculated with an algorithm called **Cyclic Redundancy Check** (**CRC**) and both FCS and CRC are used interchangeably as names for the Ethernet frame trail.

Additionally, each frame is also prefixed with a preamble (usually 7 Bytes) and **Start-of-Frame Delimiter** (**SFD**, 1 Byte). Those two do not carry additional information, however are used for initiating transmissions, a process detailed in the next paragraph. Finally, there is an additional requirement placed on the time between the last and first bits transmitted for consecutive frames, called **Inter Frame Gap** (**IFG**). Usually, the IFG represents the transmission duration of 12 Bytes. Summing up all these fields, it is formed what is called the Ethernet overhead. Namely this overhead is all network capacity that cannot be used for transmitting data.

The data on a network using shared media Ethernet is transmitted based on a **Carrier Sense Multiple Access** with **Collision Detection** (**CSMA/CD**) approach. For more details about how data is transmitted on CSMA/CD and about CD and avoidance, the reader is redirected to [15]. Furthermore, CSMA/CD was a big innovation at the time by having the sensing step and by the ability "to quickly detect the collision and abruptly stop transmitting" [83]. Still, due to multiple end stations using the same collision domain, the transmission in shared media Ethernet is neither reliable nor deterministic over time.

3.2 Switched Ethernet

A notable advancement towards a deterministic communication is in 1997, when it is introduced the switched Ethernet [15]. Each station is connected now to a port of a switch via full-duplex links. This means that in switched Ethernet, there are two collision domains on each direction between a station and a switch. Comparing to the hub, the switch forwards the data only to the machines that are interested [83]. The switched architectures are also more flexible in regard to topologies,

having now star, tree, bi- or n-torus, mesh, or hybrid topologies. Figure 1 shows a star network topology of 3 ESes and 1 switch.

With the evolution to switched Ethernet and full-duplex links, the collision problem was eliminated. The underlying issue of contention for link-capacity was shifted from the end stations to the switching nodes, where frames are buffered while waiting for the link to be free. This happens when a link is over-subscribed.

Let us reconsider the switch example from Figure 2 with the simplification that there is a single priority queue and there are no traffic shapers at egress. The transmission mechanism in switched Ethernet can be summarized as the following: whenever there is data available for transmission and the egress port senses the channel free, it forwards the data. Though, if frames of data arrive to an egress port while it is transmitting, the newly arrived frames are buffered. The contention takes place if there are at least two ingress ports sending data with a rate greater than the half of link speed. Then we say that the egress port is over-subscribed [15].

This link over-subscription causes buffer overflow, and the excess data is discarded. Even though full-duplex switched Ethernet greatly improves the overall data capacity, the upper bounds of reliability and timeliness are still infinite. Therefore, neither full-duplex switched Ethernet is suitable for TC applications.

3.3 Strict Priority Ethernet

Alongside Ethernet, the **Local Area Network** (**LAN**) suite of protocols, headed by the IEEE 802.1, was also evolving. For example, the IEEE 802.1D-1993 introduced the concept of filtering services, which allow addressing a group of stations. Furthermore, the 1998 revision of IEEE 802.1D included, among others, the mechanisms to provide QoS guarantees for certain classes of traffic. Those mechanisms start with classification, which is a process of inspecting frame headers and assigning them to classes. Classified frames are then queued into separate queues, which are in turn serviced with different priorities. Those mechanisms were further refined and extended in IEEE 802.1Q-2005 [29]

IEEE 802.1Q introduces the concept of **Virtual Local Area Networks** (**VLANs**). The process of associating within a bridge the address of a host to the serving port is called the Ethernet learning. Furthermore, this learning mechanism uses flooding, which together with other protocols reliant on broadcast, imposes that bridges forward such traffic. This in turn limits the capacity of the network available for usual unicast transmissions and causes poor scalability of LANs [83, Section 4.8.5]. VLANs aim to alleviate those limitations, by allowing users to define sub-sets of ports (on a single LAN bridge) that send each other broadcast messages. This is sometimes refereed to as *broadcast domain*.

IEEE 802.1Q adds 4 octets in the Ethernet frame header: A **tag Protocol Identifier** (**TPID**) (2 octets) and a **Tag Control Information** (**TCI**) (the remaining 2 octets). For the VLAN header defined by 802.1Q, the value of TPID is set to 8,100. TCI is further composed of **Priority Code Point** (**PCP**), **Drop Eligible Indicator** (**DEI**), and a **VLAN IDentifier** (**VID**). The PCP field spans 3-bit that specifies the frame priority (having thus available up to 8 priorities). DEI is a 1-bit flag which, depending on frame criticality, indicates whether or not a frame can be dropped in case of congestion. The VID is a 12-bit field that identifies the VLAN.

The standard classification process uses the PCP field to map incoming frames to queues. With strict priority scheduling, the transmission mechanism is as follows: if the channel is free and there is traffic available for transmission, the traffic is forwarded from the queue of the highest priority (priority 0). In this context, at each egress port, there is a queue for each priority level. The drawback is, that the lower priority traffic (lowest priority 7) waits until all higher priority traffic that is available for transmission is sent.

3.4 Avionics Full-DupleX Switched Ethernet (AFDX)

In parallel with IEEE 802.1Q-2005, evolved another protocol, namely AFDX for Switched Ethernet, proposed by Airbus and standardized by **Aeronautical Radio, INCorporated** (**ARINC**). AFDX extends IEEE 802.3 Ethernet and is the data network that can provide bandwidth guarantees and bounded end-to-end delay for flight-by-wire applications. The protocol is specified in ARINC664, part 7 (first edition proposed in 2003 [70] and standardized in 2005 [3]).

As the name suggests, the development of AFDX is driven by traffic requirements in aerospace area. The main goal of AFDX is to provide deterministic onboard communication while reducing cost and weight of the network. The initial approach in avionics was that each flight-by-wire functionality was implemented on its own device. Even more, these devices were using point-to-point connections. To adapt to ever-increasing number of applications, there are introduced the **Integrated Modular Avionics** (**IMA**) modules, represented by an ES in our notation, where multiple functionalities are sharing the same computational unit. The costs and weights of flight-by-wire systems are further reduced, while reliability is preserved, by using AFDX. AFDX replaces the point-to-point connections and introduces the virtual circuits between sub-modules of IMAs.

For ensuring QoS, AFDX has been introduced the concept of **Virtual Link** (**VL**). A VL is a directional tree-like structure, which connects one ES with one or multiple ESes. To identify the belonging VL, a frame uses a VL identifier instead of destination address. In addition to multi-cast addressing, AFDX offers the bandwidth guarantees through VLs. The bandwidth is limited for a VL by associating the maximum length L_{max} and a Bandwidth Allocation Gap BAG to each VL.

The L_{max} represents the maximum number of octets that can be transmitted at once, and BAG represents the minimum time allowed between two consequent frames belonging to the same VL. Furthermore, the BAG is the number of milliseconds restricted to powers of 2 in $\{1, \ldots, 128\}$. However, the overall bandwidth of VLs that are sharing the same link cannot exceed the 100 Mbps, the maximum speed available for AFDX. Since timing requirements are ensured by means of bandwidth limitation, we will call from now on this type of traffic **Bandwidth-Shaped** (**BS**).

In a switch, the transmission availability of a frame of the flow f_i is given by (1) the belonging VL_i and (2) the queue availability. On the first hand, for each VL VL_i, the traffic is regulated such that at most one frame is transmitted in the interval BAG^i. On the other hand, in AFDX the queues use the **First-In, First-Out** (**FIFO**) order for transmitting the traffic, therefore a frame must wait until all frames that are queued up ahead are transmitted.

3.5 Time-Triggered Ethernet (TTEthernet)

Few years later, the awareness about the need of having a communication with tighter timing and reliability constraints materializes with the standardization of TTEthernet in 2011 [73]. It extends the AFDX protocol and uses the clock synchronization mechanism that is specified in SAE 6802 [73] to provide deterministic time-critical services for applications with different criticalities. Even if development of TTEthernet is triggered by automotive area, it is used in multiple domains such as automotive, aerospace, and industrial automation. For carrying the data, TTEthernet uses the standard IEEE 802.3 Ethernet frame format.

Now, let us dive a little bit more in the historical evolution of TTEthernet. Until years 2000s, the predominant practice was to release the communication based on internal or external events. Therefore, in all areas, even in those with safety-criticality requirements such as automotive, industrial automation, and aerospace, the communication was Event-Triggered being used protocols such as CAN, Fieldbus, and SAFEbus, respectively. The main mean of controlling the critical traffic was by prioritization and limitation of used bandwidth, therefore, we will further call this type of critical traffic BS. To alleviate the timing limitations, in 1991, Hermann Kopetz provided an

academic comparative analysis of event-triggered and **Time-Triggered** (**TT**) approaches in terms of predictability, resource allocation, and system scalability[54]. The analysis is followed in 1993 by the TTP [56].

For TT paradigms, the flows of data are sent based on schedule tables that are determined a priori and installed in network nodes. In the literature, the trend is to call this type of traffic TT traffic, but since the elapse of certain amount of time can be seen also as an event; we will use from now on the term **Timely-Shaped** (**TS**) to denote the traffic that is sent at predefined points in time. Furthermore, to have a global view of the time, the clocks of the nodes that support TS traffic have to be synchronized across the network. According to authors in [54] and [55], the architectures that can support TS traffic provide a more predictable communication, but the networks where communication is driven by events have lower installation costs and resource overprovisioning.

As distributed systems start to grow in size and complexity, about years 2000s, the "mixed-criticality" architectures were introduced. In such architectures both type of traffic, BS and TS, are sharing the same infrastructure. Thus, the academic results started in 1991, drove about 20 years later to the publication of TTEthernet protocol [72, 73].

In TTEthernet, the traffic types are called classes and we can have three traffic classes, TS class, **Rate-Constrained** (**RC**) class (equivalent of BS traffic from AFDX), and **Best-Effort** (**BE**) class. At each egress port, each TS flow has its own queue. As mentioned before, frames belonging to TS class are sent and received periodically at predefined points in time. This makes TS traffic suitable for applications with delay and low jitter requirements.

Comparing to RC traffic, for which the BAG values are limited to eight powers of two, TS flows can have arbitrary periods. Furthermore, frames of RC class are sent when there are no TS frames scheduled for transmission and it is the head of the queue. Therefore, RC traffic is still suitable for applications with more relaxed timing requirements; if the bandwidth of RC flows is carefully decided across the network, there can be provided upper bounds of end-to-end delay of RC frames. BE class is for regular Ethernet traffic, and there are no delay or jitter guarantees for this class of traffic.

3.6 Audio-Video Bridging (AVB)

Simultaneous with the development of TTEthernet, is the development of suite of IEEE 802.1Q amendments colloquially known as AVB [37]. After the publishing of IEEE 802.1Q-2005 revision, the focus of AVB task group was on specifying mechanisms of ensuring communication for applications with low delay and high reliability requirements. Therefore, in 2011 these amendments are merged in IEEE 802.1Q-2011 revision [33]. For example, IEEE 802.1Qav [30] is the amendment that specifies the mechanisms of **Forwarding and Queuing for Time-Sensitive Streams** (**FQTSS**). FQTSS can be seen as a combination of switched Ethernet with strict priority and AFDX.

The synchronization of audio and video signals in very large music halls drove the development of AVB. For example, with the advent of AVB, TC audio-video, and control applications can share the same network. The main goal of FQTSS is to ensure resource availability for higher priority traffic and to prevent the starvation of lower priority traffic by limiting the bandwidth for each priority.

The traffic from a priority queue is regulated using a **Credit-Based Shaper** (**CBS**). Comparing to AFDX, in AVB the traffic is regulated on a per-queue basis rather than per-flow basis. In IEEE 802.1Qav, the traffic can be differentiated as TC (or AVB) and non-critical (or BE). Furthermore, the critical traffic can belong to AVB class A (higher priority) or AVB class B (lower priority). The availability of an AVB queue is determined by CBS. Hence, an enqueued AVB frame is available for transmission if the CBS allows it and there are no other higher priority AVB frames available for transmission.

Briefly, the CBS availability is defined as following. CBS makes the queue available for transmission whenever the amount of credit is non-negative. The credit is initially zero, it decreases with the sending slope (while transmitting) and increases with the idle slope (when frames are waiting for transmission). When credit reaches a negative value, the next frame waits until credit becomes again non-negative. If the queue becomes empty, the credit is reset to zero (for positive credit) or increases with the idle slope until reaching zero (for negative credit). the idle slope represents a fraction of link speed and the sending slope as difference between idle slope and link speed. Different QoS is ensured for critical traffic by using a higher idle slope for class A comparing to class B.

3.7 Time-Sensitive Networking (TSN)

One year later, in 2012, the IEEE 802.1 working group was renamed to TSN to show that the interest now is towards TC applications in general. TSN purpose is to extend the IEEE 802.1Q VLAN protocol to meet the timing and reliability requirements of distributed applications. TSN rather than being a single new communication protocol, is a set of tools and mechanisms that can be applied on top of IEEE 802.1 standards. Those mechanisms are first described in amendments to the aforementioned standards, and then incorporated into next revisions.

If the development of timing-deterministic protocols was triggered previously by needs in different areas, TSN is built from the beginning with the general applicability in mind. TSN followed from the beginning the trend towards standardization. Before that, the approach was that each area used specific protocols, such as CAN and FlexRay for automotive, SAFEbus and AFDX for airspace industry, and field-buses, ProfiNet and EtherCAT for industrial automation. The aim regarding standardization is that protocols for TC and reliability are implemented in COTS networking components. This implementation leading thus to a market competition for vendors, which will determine for COTS the increase in quality and the decrease of prices.

For addressing in TSN the timing constraints, traffic control mechanisms are added, such as **Time-Aware Shaper** (**TAS**) as specified in IEEE 802.1Qbv [39] (Enhancement for Scheduled Traffic) and IEEE 802.1Qbu [41] (Frame Preemption). The transmission of scheduled traffic requires that clocks of network devices have the same notion of time for this being defined the clock synchronization mechanism in IEEE 802.1AS [47].

The resource allocation is specified by **Stream Reservation Protocols** (**SRPs**) IEEE 802.1Qat [31] (Stream Reservation Protocol), IEEE 802.1Qca [40] (Path Control and Reservation), and IEEE 802.1Qcc [45] (SRP Enhancements and Performance Improvements) for instance. The fault-tolerance requirements are addressed for faulty network components by IEEE 802.1CB [42] (Frame Replication and Elimination for Reliability) and for bursty traffic by IEEE 802.1Qci [43] (Per Stream Filtering and Policing). These are just some of standards and amendments that compose TSN, for an exhaustive list, please see IEEE TSN page at [38].

As for previous protocols we were concerned with traffic forwarding, in the following we will present briefly how queuing and forwarding works in IEEE 802.1Qbv. To ensure low delay and jitter for scheduled traffic, The TAS is specified in IEEE 802.1Qbv. TAS imposes independent time windows by opening and closing the gates that are associated to each queue of an egress port. The opening and closing times are stored in the so-called **Gate Control List** (**GCL**). Delays induced by lower priority traffic is prevented by closing the gates of respective queues. When the channel is free, the next frame selected for transmission has the following properties: (1) the queue's gate is open and (2) belongs to the queue with the highest priority. The full control of forwarded traffic is provided by queues' gates opening in a mutually-exclusive way.

Furthermore, in TSN, the scheduled traffic can co-exist with AVB and non-critical traffic. When AVB shares a link with scheduled traffic, the credit of AVB classes is frozen during the timing windows allocated for scheduled traffic. Moreover, when scheduled traffic share the infrastructure

with (Possible non-critical and) non-scheduled traffic, it might be delayed by AVB or BE traffic already transmitting. TSN uses traffic integration modes which are mechanisms to reduce this type of delay. These modes are guard band and preemption. The guard band mode is already detailed in our previous work [23] therefore, we will not insist on it.

The other integration mode is preemption, which is standardized in IEEE 802.1Qbu [41] extending the IEEE 802.3br [35] which introduced the concept of express traffic. The mechanism for one-level preemption is that the transmission of a frame that belongs to preemptable traffic is interrupted by the transmission of an express frame. A common configuration is to consider that in TSN the express traffic is the one scheduled. Furthermore, the transmission of preemptable frame is resumed after the scheduled transmission. As mentioned by the amendment, the fragments of preempted lower priority frames are forwarded in well-formatted Ethernet frames, i.e., the so-called smallest non-preemptable fragment carries at least 64 Bytes of data. Therefore, the transmission of a scheduled frame is delayed in the worst-case by the transmission duration of the smallest non-preemptible fragment. Though, to fully cancel this delay and to reduce the waisted bandwidth, the guard band and preemption integration modes can be combined.

3.8 Summary

This sub-section presents an overview of deterministic Ethernet-based communication protocols, highlighting the advantages and disadvantages of each. The first versions of Ethernet, such as shared media Ethernet, switched Ethernet, and strict priority, could not provide any timing and reliability guarantees. But, as we could see, these first versions of Ethernet were necessary steps towards deterministic protocols.

Let us start with AFDX whose benefits are that (i) it provides bounded end-to-end delay, (ii) the guarantees are ensured on per-flow basis, (iii) it is easy to deploy; the values for BAG and L_{max} can be determined based on flow properties, and (iv) it does not require the clock synchronization of network nodes. On the other hand, the main drawbacks of AFDX are (i) the jitter experienced by critical traffic, (ii) the high **HardWare** (**HW**) implementation costs; on each network node there is a buffer for each VL, (iii) the timing analysis method required for computing the WCD of flows, and (iv) the values for BAG are restricted to 8 powers of 2.

Going further, TTEthernet obliviates some of the drawbacks from AFDX (regarding jitter and need for timing analysis for example) but it has some disadvantages of its own. In a nutshell, the advantages of TTEthernet are that (i) besides bounded end-to-end delay, it can provide 0-jitter due to contention to TS traffic, (ii) similar with AFDX, the guarantees are ensured on per-flow basis, (iii) the flows' priorities can take arbitrary values, (iv) it has support for mixed-criticality applications, and (v) it does not require timing analysis method for computing WCD of TS traffic. TTEthernet innate drawbacks are that (i) similar with AFDX, it has high HW implementation costs; on each network node there is a buffer for each VL of a BS or TS flow, (ii) it requires scheduling method for computing the schedule tables for TS traffic across the network, and (iii) it requires the clocks of network devices to be synchronized.

AVB reduces the HW implementation costs, as experienced for AFDX and TTEthernet, by ensuring deterministic requirements on a per traffic-class basis. Similarly, with AFDX, AVB (a) provides bounded end-to-end delays, (b) is easy to deploy; the profile for audio-video systems IEEE 802.1BA gives default values for idle slopes, (c) is not able to limit the jitter, and (d) requires a timing analysis method for computing the WCDs of BS traffic. In contrast with AFDX, it needs the clocks of switches to be synchronized in order to estimate the end-to-end delay of a stream during resource reservation. This need for delay estimation during resource reservation means that each switch decides, based on this estimation, if to forward a stream or not, only if by doing so the timing

requirements for this stream can be met. In addition to AFDX, AVB benefits of higher network speeds, being able to ensure an end-to-end delay of tens and hundreds of microseconds.

The last, but from farthest the least, of the protocols is TSN, particularly TAS, which combines the advantages of TTEthernet and AVB. Briefly, the benefits of TSN are as following: (i) it can ensure bounded and low end-to-end delay and jitter (as in TTEthernet), (ii) HW implementation costs are reduced by using per traffic-class scheduling and shaping (similar with AVB), (iii) it provides support for mixed-criticality applications (as in TTEthernet), (iv) it hinders vendor and area lock by mean of standardization, and (v) on fully deterministic mode, does not require timing analysis method for computing WCD of TS traffic (similar with TTEthernet). The most important downsides are that (i) when using IEEE 802.1Qbv, it is required a very complex method for computing the schedule tables of TS traffic across the network (similar with TTEthernet) and (ii) it requires that the clocks of network nodes are synchronized (as in TTEthernet and AVB).

4 DESIGN OF TIMING-CRITICAL NETWORKS — PROBLEMS' FORMULATION

In this section, we are presenting the most important tasks for designing a TC network. Our listing is based on the overview regarding design tasks for TC systems from [72]. Because our focus is on ensuring reliable and TC communication, we will address the design tasks related to topology selection and synthesis of configuration parameters that are influencing the traffic transmission. Therefore, the design tasks that are further defined are: (1) network planning and design, (2) routing, (3) priority/traffic type assignment, (4) scheduling, and (5) bandwidth allocation. Based on the SDN nomenclature established in the introductory section, the first two design problems fall under the control plane while the later three design problems fall under the management plane.

The goal of this section is to formally define the tasks of designing networks that have to support also TC distributed applications. As size, complexity and usage of distributed systems grow, so need to evolve the methods to design such systems. Therefore, **Computer Aided Design** (**CAD**) is used more and more for the engineering of large-scale systems [17]. To help the development of CAD solutions for designing TC networks, through this section, our aim is to highlight the complexity and requirements of each task.

After this brief overview of the design tasks, we are going to formalize in the following paragraphs each design task as a networking design problem. Based on the network and traffic models introduced in Section 2, for each problem, we are introducing the input, output, and constraints of the results. Regarding the validity of a solution, in addition to traffic requirements that have to be met, we will highlight the conditions in which a solution is correct. Moreover, for each type of problem we are highlighting some optimization objectives that can be formulated.

4.1 Problems' Definitions

Network planning and design. For a given set of ESes $\mathcal{ES}$ and traffic model $\mathcal{F}$, the aim of topology selection problem is to determine the logical topology of the network $G(\mathcal{N}, \mathcal{E})$ such that the traffic can be successfully forwarded. When deriving the network topology, the available network components, such as NSes and links, are provided through the so-called CL. For each component, it is considered known the type (set of features) and cost. Furthermore, Determining the logical topology means to find the number and type of NSes $\mathcal{NS}$ and links $\mathcal{E}$ and how to interconnect the ESes and NSes.

Let us recall that a flow of data f_i, among other things, is characterized by a source ES and possible multiple destination ESes. Therefore, for a valid network, it is required that for all flows $f_i \in \mathcal{F}$ the pairs of source-destination ESes are connected with at least one path. Let us recall that in TSN a switch is called also bridge, which means that when building a network, no switches

should be end points. This constraint means that all switches must have at least two connections. These are hard constraints that any valid network has to fulfil.

When building a network, usually there are defined extra objectives that have to be considered. For example, the most common objective is to minimize the monetary cost of the network. For airspace and automotive areas, a similar objective is to minimize the weight of cabling and network devices. The objectives though are dictated by the requirements of the distributed applications that are accommodated by the network. For example, in switched Ethernet, the link congestion and end-to-end delay can be controlled only by mean of network planning and design. Furthermore, for later protocols, if the network engineer decides that some fault-tolerant applications have to be routed through disjoint paths, the network should be designed such that these kinds of paths can be created.

Routing. A problem closely related to the topology selection, is the route computation problem. Giving the network topology $G(\mathcal{N}, \mathcal{E})$ and the set of flows $\mathcal{F}$ including their properties and requirements, the task is to determine the route r_i over which the data of each flow $f_i \in \mathcal{F}$ is forwarded such that requirements of the applications are met. As mentioned in Section 2.2, a route represents a DP and tree for unicast and multicast flows, respectively.

As the route is the structure that connects the source ES with the destination ESes, a valid route should satisfy several hard constraints. One of the constraints is that all destinations should be reached. Another constraint is that a route should be cycle-free. In addition to these constraints, while modeling a routing problem, the network engineer can be interested to optimize also different objectives, such as minimizing the number of spanned links or minimizing the overall weight of the links. For the latter case, the weight of a link usually represents the used bandwidth or the link propagation delay, but in general it depends highly on the characteristics of the applications and the protocols in use.

Priority and Traffic type assignment. The goal of priority assignment problem is to decide the most appropriate priority for each flow $f_i \in \mathcal{F}$ for the given set of data flows $\mathcal{F}$ and the maximum available priority. The priority must be assigned such that the requirements of the critical traffic are satisfied. Initially, to differentiate among critical and non-critical traffic, priorities were introduced, i.e., when competing on an outgoing port higher priority traffic takes precedence over lower priority traffic. The priority assignment technique is more suitable for networks based on protocols such as strict priority Ethernet, AFDX, and AVB.

Furthermore, the distributed applications have been used more and more in safety-related areas and their complexity escalated in size and requirements. Therefore, to meet the new requirements, different traffic types were introduced to dictate how flows are shaped on network devices outgoing ports. We would like to recall that the main split is between TS and BS types of traffic. We would like to highlight that there is no one-to-one mapping between priorities and traffic types. In general, in the literature, it is considered that the TS traffic has the highest priority, the BS traffic has a lower priority, and that non-critical traffic is associated with the lowest priority. Though, there are cases where multiple priorities are considered for a single traffic type, e.g., multiple priorities can be considered for BS traffic in TTEthernet and TSN. Moreover, if QoS is provided also for the non-critical traffic, multiple priorities can be considered for this type of traffic.

As suggested before, the problem of traffic type (and priority) assignment is encountered for networks that are based on TTEthernet and TSN protocols. Depending on each type of traffic, there are additional configuration parameters that have to be decided. Namely, for the TS traffic have to be computed the schedule tables and for BS traffic have to be allocated the bandwidths. Therefore, in the following paragraphs we are going to introduce these schedule computation and bandwidth allocation problems, presenting for each of them also the constraints and objectives.

Scheduling. This problem deals with deriving the schedule tables $\mathcal{S}$ for TS traffic such that all TC traffic is schedulable. For the scheduling problem, it is given the network topology $G(\mathcal{N}, \mathcal{E})$ and the traffic model $\mathcal{F}$ including all their properties and requirements along with the routes for all flows $\mathcal{R}$. The TS flows are transmitted and received at predefined points in time that are stored in structures called schedule tables. Thus, the time when the jth frame of a TS flow f_i is transmitted on link $dl_{x,y}$ is denoted $\phi^{i,j}_{dl_{x,y}}$.

To ensure a predictable transmission in general, a valid schedule table must satisfy some constraints. The first constraint is about the *link occupancy*; a link can forward at most one frame at a time. This means that all TS frames should be scheduled for transmission on disjoint time slots. Another constraint is the *flow-dependent* one; a frame should be entirely received by a switch before being forwarded on the next link. The time when a frame is fully received by a switch cannot be precisely known; even if the clocks of the network devices are synchronized, there is no perfect clock synchronization mechanism. Therefore, when building the schedule table, the forwarding time of a frame, in addition to forwarding delay on previous link, should take into account the clock synchronization error δ_{synch}. Finally, another constraint that describes a valid schedule table is the *end-to-end delay* constraint; which specifies that any frame, after being dispatched from its source, should reach its destination before the deadline.

Furthermore, in TSN, the schedule tables are represented as GCLs. Because in TSN are controlled the times when gates of specific queues are opening and closing, the computation of schedule tables has to decide upon additional parameters. Therefore, in TSN have to be decided for each switch ns_i the number of queues, allocated for TS traffic on each egress port j, denoted $Q^{TS}_{i,j}$ and the mapping of each TS flow to a queue. This mapping for a TS flow f_i on a link $dl_{x,y}$ is denoted $q(f_i, ns_x); q(f_i, ns_x) \leq Q^{TS}_x$. Furthermore, we say about a flow f_i that is schedulable when its **Worst-Case Delay** (**WCD**) is at most equal with its deadline D_i.

As mentioned before, in TSN the scheduling is done on a per-queue basis. Therefore, an additional constraint should be considered when determining the GCLs, namely the *queue occupancy* constraint. To ensure a fully deterministic transmission of TS traffic, the queue occupation intervals for TS frames should be disjoint. This means that a single frame can be present in the TS queue before and during a gate-opening event. There exist in the literature also a relaxation of this constraint, namely that the frames of a same TS flow are allowed to share the queue for a single gate-opening event. This relaxation requires though that at the destinations there are mechanisms to recover for a TS flow from a possible miss ordering in receiving of the frames. Moreover, the queue occupancy intervals should be separated by clock synchronization error δ_{synch} for queue occupation intervals of TS traffic coming from different ingress ports.

The WCD of TS flows is determined based on schedule table for networks using TTEthernet and for TSN networks where GCLs are synthesized for fully deterministic transmission of TS traffic. Otherwise, if fully deterministic constraints are ignored for TSN-based networks, the WCD should be computed using highly complex timing analysis methods. Such a timing analysis method based on network calculus is presented in [85].

When synthesizing the schedule tables, different objectives can be taken into account. For example, a network engineer can be interested to make the TS traffic to arrive as soon as possible, by maximizing the overall laxity for TS flows. The laxity of a flow is defined as the difference between deadline and WCD when the flow is already schedulable. Another objective is to minimize the tardiness of the non-scheduled critical traffic. The tardiness is defined as the difference between WCD and deadline, but this applies also for the non-schedulable flows. For a TSN-based network, the network engineer might be interested to minimize the maximum number of queues that are used for TS traffic.

Bandwidth allocation. For the given network topology $G(\mathcal{N}, \mathcal{E})$, traffic model $\mathcal{F}$ including flows' properties, requirements and routes and the set of available bandwidths $\mathcal{BW}$, the goal of this problem is to determine the most appropriate available bandwidth for the non-scheduled critical traffic such that the timing requirements are met. Depending on the used protocols, the bandwidth can be imposed on a per-flow or per-traffic priority/type basis. Furthermore, this problem can be extended to adjust also the set of available bandwidths, i.e., to modify the current bandwidths or to add other bandwidth elements. Assigning an available bandwidth to a flow, it means that the bandwidth on transmission for the respective flow is upper bounded. By bandwidth limitation, the critical non-scheduled or lower priority traffic is prevented to wait at infinity on network devices egress ports reducing thus the WCD of non-scheduled traffic. Therefore, the bandwidth allocation problem can be seen as the scheduling problem for the non-scheduled traffic.

When allocating the bandwidth, the hard constraint is that the overall bandwidth of the traffic should not exceed the link bandwidth. Furthermore, when designing a mixed-criticality network, the objectives from the scheduling problem apply also here. We would like to recall that a mixed-criticality network is one on which highly TC (and possible scheduled) traffic share the same network with the critical non-scheduled traffic and non-critical traffic. Therefore, when deciding for the bandwidth the objectives can be extended to optimize for the timing characteristics also for the lower priority traffic. For example, an objective can be to reduce the bandwidth used for critical traffic as much as possible such that a certain QoS can be provided also for non-critical traffic. The bandwidth allocation is more suitable for networks that are using AFDX and AVB protocols. Though, an interesting approach is to use the bandwidth allocation technique also for TTEthernet and TSN protocols. This technique can be used either when schedule tables are fixed or it can be a joint technique to optimize the bandwidth for BS traffic and schedule tables for TS traffic.

4.2 Evaluating End-To-End Delay of BS Traffic

The average end-to-end delay of BS traffic can be computed using methods of queuing theory. A queue that forwards BS traffic can be modeled, for example, as an M/M/1 queue: first "M"; means that BS traffic arrival follows a Poisson arrival process, second "M"; traffic from the queue is forwarded following a Poisson distribution and "1"; there is a single server for the queue. Furthermore, for this model, the average arrival rate of the traffic is denoted λ and the average service rate is denoted μ. Please note that in literature the service rate is named also service time. Furthermore, the degree of queue utilization is denoted $\rho = \frac{\lambda}{\mu}$ and it is computed as the ratio between arrival rate and service rate. Thus, the average waiting time w_t for a packet in the queue is estimated by the Pollaczek–Khintchine formula as $w_t = \frac{\rho}{\mu(1-\rho)}$.

However, the WCD for BS traffic can be obtained only by using complex timing analysis methods. There have been several WCD analyses proposed for BS traffic on AFDX, such as the work from [19] based on network calculus, methods in [7, 8] a network calculus and trajectory approach comparison and trajectory approach techniques presented in [9]. The researchers have proposed WCD analysis methods in TTEthernet for BS traffic that have to co-exist with TS traffic. Such methods are [82] (based on trajectory approach) and [86] (using network calculus).

Furthermore, for AVB-based networks there exist timing analysis methods to compute the WCD for BS traffic [14] (based on network calculus). A technique studied more thoroughly is though the one of timing analysis for BS traffic in presence of TS traffic for networks based on TSN. Initially, the AVB latency Math equation from the AVB profile IEEE 802.1BA has been extended in [57] to consider the TS traffic. However, this method is quite limited; it handles only AVB Class A traffic and, according to authors in [87], the analysis is both unsafe and overly pessimistic. Later on, were proposed also other methods such as [59] (An availability interval-based approach) and [87] (a network calculus approach).

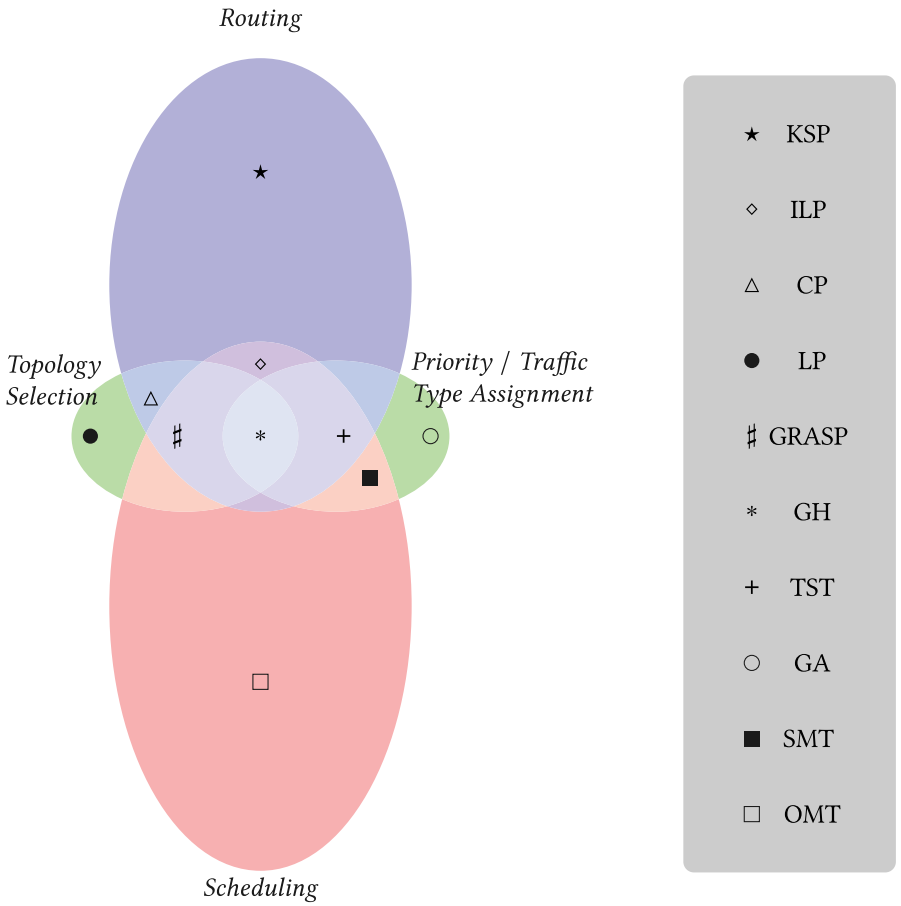

Fig. 4. Overview of strategies used for types of design problems; each strategy has its own symbol and types of design problems are represented as different ellipses.

5 DESIGN OF TIMING-CRITICAL NETWORKS — STRATEGIES

This section makes an overview of the strategies that exist in the literature about designing networks that have to support TC distributed applications. Such applications are used more and more in areas where timing predictability is of outmost importance. Therefore, the methods for designing networks that can support such applications should evolve as well. But to drive a real evolution, we consider that is important to have an overview of the methodologies and results obtained so-far in the area in order to avoid reinventing the wheel. Although, design strategies exist also for switched Ethernet, AFDX, and TTEthernet, we are focusing on strategies proposed for TSN-based networks.

Figure 4 represents a graphical overview of the strategies mostly used in literature for designing time-aware networks. The types of problems are represented by ellipses and types of strategies with different symbols. For sake of readability, we used in the figure the abbreviation of the strategy names, and we introduce here their descriptions: CP (Constraint-Programming), GA (Genetic Algorithms), GH (Greedy Heuristic), GRASP (Greedy Randomized Adaptive Search Procedure), ILP (Integer-Linear Programming), KSP (K-Shortest Path), LP (Labeling Problem), OMT (Optimization Modulo Theory), SMT (Satisfiability-Modulo Theory), and TSt (Tabu-Search Technique).

The goal of this article is to establish a terra firma of the methods that are already published for the area of designing TC networks. In general, the search for near-optimal solution is computationally intractable. Therefore, in this work, we present heuristic (constructive) and metaheuristic (optimization) strategies for design-space exploration [10].

For the problems listed previously in Section 4, we will present some of the most relevant strategies presented in the literature including also a brief discussion about the results. For each problem we will show first the constructive strategies (when available) and afterwards the optimization strategies that were proposed. The strategy name will be highlighted in italic fonts. Furthermore, when relevant, the strategies will be presented in terms of traffic and network models, of protocols, of considered constraints and quality and of obtained results. When applicable and available, for the input, we would like to record the following metrics:

— Size and/or the monetary cost of the network
— Network speed (in Mbps)

For this work, to compare two solutions from the design-space we use the so-called quality function of a solution. Worth noticing is that the quality function is a general term that might not reflect the validity of a solution. Furthermore, when analyzing the output, we will target the following metrics:

— Timeliness-related results;
Are the strategies able to improve the end-to-end delay and jitter for the critical flows? Furthermore, are the results presented for the worst-case or only for the average case?
— Throughput-related results;
which is the maximum percentage of, optional critical, traffic that can be handled by the strategy?
— Optimality-related results;
how far from the optimal is the found solution?
— Strategy performance-related results;
what is the running time of the strategy and can the strategy be used for online configuration?

We would like to highlight that there is no clear line between given and obtained parameters; this separation depending on the type of problem. For example, for the routing problem, the bandwidth usage is determined, but for the priority/traffic type assignment problem and scheduling problem it is given. Though, we can state that the number of ESes must be always given and that the optimization- and runtime-related metrics are always obtained.

5.1 Network Planning and Design

The problem of network planning and design was already formalized in the previous section. Here we are going to present design strategies proposed for this problem in the literature for TSN-based networks. Please note that throughout this article, we are also using the terms topology selection and topology synthesis for denoting the network planning and design problem.

The topology synthesis problem, driven by routing, is addressed for TSN-based networks in our previous work from [22], for fault-tolerant but non-scheduled traffic. The same problem, but driven by routing and scheduling, is addressed though for TSN-based networks in [6]. The work in [6] assumes that the CL consists of multiple types of switches and a single type of links. Furthermore, this work assumes that the redundancy level is given, to indicate how many disjoint paths are required for a critical flow.

In the work from [6], the authors formulate the topology synthesis problem as a labeling problem. The joint topology selection, routing and scheduling problem is solved (1) by computing an initial maximum topology, (2) by using the *KSP* routing for reducing the search space for routes, (3) by applying the *least scheduling* heuristic for computing the schedule tables, and (4) the routing and scheduling configuration is selected using a GH. The technique assigns for each flow (a) iteratively the KSP and (2) available time slots for scheduling on the selected route. The adaptive aspect of the technique is that on each iteration, after the route for a possible redundant flow is selected, the weight of each arc is updated to reflect the newly supported traffic.

To evaluate the adaptive procedure in [6], there are conducted 2 sets of experiments: first for evaluating the monetary cost and the percentage of feasible solutions and second the runtime of the strategy. For the first set of experiments the results of the adaptive strategy are compared with other two strategies: duplication of network for obtaining fault-tolerant transmissions and our optimization solution from [22]. As expected, the strategy of duplicating the network gives the costliest solutions. Though costly, the network duplication finds feasible solutions in 90% of cases with the largest number of flows. Even if the optimized strategy from [22] gives the solutions with the smallest network cost, the evaluation in [6] highlights the importance of choosing the transmission model and how much the network size can affect the schedulability. As we can observe, the adaptive strategy from [6] renders 100% feasible solutions in all cases.

Furthermore, the second set of experiments from [6] evaluates the runtime of the adaptive strategy for different input sizes. Even if the authors claim that the adaptive strategy scales well with the size of input, we consider that the strategy scales well only in terms of traffic load. A network with 24 stations is not that big and as it can be observed from the reported results, the runtime does increase rather exponentially regarding number of stations.

5.2 Routing

As it can be observed, the problem of routing was already tackled jointly with the problem of topology selection. However, the problem of choosing the routes in order to improve the timeliness has been addressed also in isolation or jointly with other design problems. As already introduced in Section 4, the routing problem takes as input the network topology $G(\mathcal{N}, \mathcal{E})$ and the set of flows $\mathcal{F}$. We would like to mention that in general the objective for the routing solutions is to minimize the maximum link usage.

5.2.1 Routing for TSN-based Networks. The problem of finding the routes for AVB flows over TSN-based networks is addressed in [57]. This work presents the routing solution for TSN networks that have to support mixed-criticality applications with timing constraints, i.e., the deadlines of TS flows have to be met and the WCDs of BS flows have to be minimized. The work in [57] considers that the network topology is given as well as the routes and schedule tables for TS traffic. To solve this problem, the authors use a *KSP* method to reduce the search space of routes and a *GRASP* metaheuristic for optimizing the routing.

Here, a routing solution is considered valid when all AVB flows meet their deadlines. As mentioned in Section 4, to compute the WCD of an AVB flow, this work uses the AVB latency math defined in the profile for audio-video IEEE 802.1BA [32]. In [57], the AVB latency equation is adapted to account for TS traffic. The objective of the problem in [57] is to minimize both WCDs and network utilization for AVB flows. Furthermore, to evaluate a routing solution, the function is composed of the schedulability check and the objective. This means that the authors are interested to find the routing solution which renders all AVB flows schedulable and gives also the smallest overall WCDs of AVB flows.

Another solution to determine the routes of TS traffic for TSN-based networks is presented by authors in [63]. The problem is to find the routes of the TS flows such that the slack is maximized. First of all, we would like to mention that this work assumes a no-wait transmission model that is detailed in [16] and will be described further in Section 5.5. In this scheduling model, the tendency is to schedule frames back-to-back. Therefore, the slack is the amount of time available for non-scheduled traffic (excluding also the guard band required for transmission of TS traffic) within a scheduling cycle. The idea is that by distributing the transmission of TS traffic over the network and maximizing the slack, it is maximized also the time available for transmission of non-scheduled traffic.

To solve the routing problem, the authors in [63] model it as an *integer linear problem*. Worth noting is that for addressing the multi-cast routes, the structure destinations count *DC* is defined similarly as matrix *X* in [5]. Another interesting aspect is that the work in [63] does not constrain the routes to be cycle free, speeding thus the search process. To ensure though a cycle-free routing, after optimization, it is applied a post-processing step.

For solving the routing of TS traffic in TSN, the authors in [63] propose two approaches. For the first approach, the objective is to minimize the **Maximum Scheduled Traffic Load** (**MSTL**). The scheduled traffic load is computed as the time used by the scheduled traffic on all ports of a switch on the span of a scheduling cycle. Therefore, the MSTL for a switch represents the maximum scheduled traffic load from an entire schedule length. The second approach uses an objective that considers both MSTL and number of hops.

Furthermore, the routing of fault-tolerant and TC traffic for TSN-based networks is presented in [64]. Here, the authors attempt to reduce the delays of TC traffic by minimizing the maximum bandwidth usage on links. As input for this problem, in addition to well-known properties and requirements of flows, the replication level should be also given. This replication level is a fault-tolerance requirement which indicates through how many as link-disjoint as possible routes a flow should be forwarded.

For solving this routing problem, the authors in [64] use a load-adaptive and redundancy-aware GH to construct a utilization balanced routing solution. Furthermore, a congestion-driven *tabu search* metaheuristic is proposed to improve a routing solution by rerouting the flows that are forwarded through a congested link. In the work from [64], the greedy-based routing solution is constructed by selecting the "best" valid route from the set of all valid routes for a flow. Since it is not known the optimal order on which flows must be routed, the "best" route for a flow can led to overloaded links on later routings.

Therefore, to escape from a local optimum, the second strategy from [64] comes to improve a given routing solution by rerouting the flows routed initially through congested links. A link become congested when the used bandwidth is over a certain threshold. The second approach tags as "tabu" the congested links by removing them temporally from the network, attempting thus to avoid them. For each flow previously routed through congested links, the congestion-driven strategy calls again the route selection method.

As stated before, the objective of the constructive and optimization strategies from [64] is to distribute the link loads over the whole network. In terms of valid routes, this work is very generic. It is claimed that a valid route is one that ensures that the deadline of the forwarded flow is met, but there is no specific transmission model, hence also no specific timing analysis method to evaluate the end-to-end delay of TC flows.

Evaluation of Routing in TSN. As we can see from the results in [57], the optimization routing strategy is effective for the schedulability of BS flows. Contrary to straight-forward strategy (that uses the *SP heuristic*), the optimization strategy gives schedulable solutions in 4 out of 5 realistic

test-cases. What is very interesting to observe is the trend of the number of links used for routing the BS traffic. The number of used links slightly increases for the small synthetic test-cases by using the routing meta-heuristic. Though, using the same strategy the number of used links is decreased for the larger realistic test-cases, with a maximum decrease of about 7%.

Next, for evaluation in [63], there are compared different routing strategies. The first two are some available-out-of-shelf strategies, namely the **Shortest Path** (**SP**) and **Equal-Cost Multi-Pathing** (**ECMP**). The other two approaches are the two ILP formulations with the two objectives: the minimization of MSTL and the minimization of MSTL plus number of hops, respectively. However, for all the four strategies are evaluated the MSTL, the flowspan, and the runtime. The concept flowspan is better described in [16] as the latest time when a flow, among all TS flows, reaches its destination. In the following, we refer as an increase factor of 1 when doubling the value on X axis the value on Y axis is also doubled. Therefore, the first measurement, always gives a factor of 1 since it is the baseline.

As the results of the first set of experiments in [63] show, the MSTL and flowspan increases regarding number of TS flows, when using SP strategy, linear but much steeper concerning flowspan. The linear increase has an average factor of 0.77 (for MSTL) and 1.09 (for flowspan) and a maximum factor of 1 and 1.4, respectively. Furthermore, since ECMP attempts to spread more the TS traffic over the whole network, the MSTL increase is much slower, observing even a sub-unitary and decreasing trend of increase factor after 600 TS flows. Therefore, by using ECMP, we can observe an increasing average factor of 0.83 (for MSTL) and about 0.81 (for flowspan) and a maximum factor of 1 for both metrics. Regarding the two ILP formulations, for the maximum number of flows, it can be observed that ILP with MSTL objective renders the same result as ECMP and that only ILP with MSTL + number of hops objective more than halves the flowspan comparing with SP.

The second set of experiments from [63] evaluates how much the ILP formulations manage to reduce the flowspan comparing with SP and ECMP routing strategies. As the figures show, the MSTL oriented ILP formulation is not that stable, being able to decrease, comparing with SP, the flowspan with about 40% only in 6 out of 24 cases, a decrease of 20% being observed in 18 out of 24 cases. Comparing now with MSTL and number of hops-based ILP strategy, it highly outperforms SP and ECMP strategies. For example, the ILP decreases the flowspan, comparing to SP, with 20% in almost 22 out of 24 cases.

In our opinion, the most interesting results can though be observed for the ILP formulations concerning the running time. Let us recall that the difference between the two ILP formulations are the objectives. If the first formulation considers only MSTL minimization, the second one uses the minimization of number of hops and MSTL. As we saw previously, the second ILP approach is able to reduce the flowspan, but not that spectacularly. Though, the runtime of second formulation increases with 2 orders of magnitude comparing to the runtimes rendered by only MSTL-based approach. As this third set of experiments from [63] highlight, ILP formulations are very sensitive to the complexity of objective formulations.

Finally, as it can be observed from the results in [64], the load-adaptive strategy reduces the maximum bandwidth used over a link comparing to SP and ECMP strategies. Though, as we can observe from the reported results, the load reduction regarding the number of flows is not that spectacular as claimed by the authors. Namely, the load-adaptive reduces the load for the worst-case scenario (with 200 flows) with 18% comparing to maximum load obtained with SP algorithm. Though, from the reported results, it is clear that an increased network connectivity helps load-adaptive strategy to find better routing solutions.

In terms of runtime, the load-adaptive strategy from [64] does not perform that well, even if there is no time spent for evaluating the WCD for the routing solutions. Surprisingly, the runtime of strategy increases only linearly with the number of flows. Though, as expected, the runtime

increases exponentially regarding the number of nodes. As the results show, the load-adaptive strategy is not suitable for runtime routing due to the too long running times for moderately large networks of 120 nodes.

As further highlighted by the authors, the congestion-driven strategy from [64] is also not a promising candidate for runtime load-balanced rerouting; following a similar trend in runtime as load-adaptive strategy. While link utilization highly varies from 32 flows to 2 flows per link with SP routing, the congestion-driven strategy is very effective in balancing the link utilization. Thus, we can observe that after congestion reduction, the mean is of 10 flows per link with a variation of 5 flows. Furthermore, by applying the congestion-driven strategy also in terms of load distribution the peak is lower comparing with SP algorithm.

5.3 Priority and Traffic Type Assignment

In this section, we are presenting separately the solutions for priority and traffic type assignment problems. As introduced previously in Section 4, for accommodating ever-increasing timing demands of distributed applications, in addition to priorities, the protocols have introduced the traffic types. These traffic types are used to indicate the relative timing requirements of the flows. Usually, the traffic types that can guarantee more stringent timing requirements have higher priorities. But as already highlighted in Section 4, within the same traffic type can exist also multiple priority levels.

5.3.1 Priority and Traffic Type Assignment for TSN-based Networks. One of the first papers that addressed, in the context of TSN, the problem of priority assignment for **Asynchronous Traffic Shaping** (**ATS**) traffic is the one in [79]. The work in [79] searches for the minimum number of queues that are required to guarantee the timing requirements of ATS traffic. As our overview paper focuses on TAS and CBS traffic in TSN, we will not dive into more details of the work from [79].

The problem of priority assignment is also addressed for the preemptive approach in TSN in [65]. As input for the problem are the set of flows $\mathcal{F}$ and the network topology $G(\mathcal{N}, \mathcal{E})$. Furthermore, each switch ns_i has a number of ports and it is assumed that for all switches from the network it is known the maximum number of priorities. Therefore, the goal in work from [65] is to assign priorities to flows and decide how many queues from an egress port of a switch are allocated for express traffic. These parameters are determined such that flows are meeting their deadlines and the schedulability and reliability are improved. To solve this problem, the authors use an GA approach.

To model the problem from [65] as a GA problem, an individual has 2 genes. The first gene consists of a vector that maps priorities to each flow. The second gene consists of a matrix Z, with $\zeta_{i,j}$ denoting the number of queues used for express traffic on the jth port of switch ns_i. To generate the initial population, there are used 3 hypotheses about what can improve a solution. Namely, by assigning more critical traffic to higher priorities, keeping a balanced load among queues of an egress port and assigning lower priorities to flows with smaller frame size. Using these hypotheses, there are 10 solutions generated for initial population. Furthermore, to shorten the comparison time of solution during the search, the individuals of the population are stored ordered regarding the fitness function.

A new generation is created by applying the crossover operator and possible the mutation operator. The crossover is done by selecting 2 individuals. The offspring is obtained by selecting the first gene from the first parent and the second gene from the second parent. The mutation operator changes randomly an element of either first or second gene. If the offspring has a better fitness function than the last individual from the population, the last individual from the population is

removed and the offspring is added to the population on the position indicated by its fitness. The search finishes after visiting 100,000 generations.

As mentioned before, besides schedulability, the strategy in [65] attempts to improve also the reliability. For this purpose, the transmission model assumes that in addition to frames transmission from source to destination, there is the transmission of the **acknowledgment** (**ACK**) frame from the destination back to source. Therefore, besides minimizing the WCD for frame f_i, has to be minimized also the WCD of ACK frame $f_{ACK,i}$. Furthermore, the reliability is increased when the **Critical Scaling Factor** (**CSF**) of a flow is maximal. The CSF of a flow represents the number of retransmissions that can be executed within a flow deadline. Please notice, that a retransmission represents the transmission of a frame and its ACK frame. Therefore, to evaluate a solution of the problem in [65], the fitness function represents the minimum CSF of the flows. Furthermore, a better solution is one which maximizes this fitness function.

Furthermore, we addressed the problem of traffic type assignment also for TSN-based networks in [21]. This solution is similar with the one presented for TTEthernet. As we already presented in Section 3 the differences between TS and BS types in TTEthernet and TSN, we will not further insist on analyzing the extra complexity of addressing the traffic type assignment for TSN. However, we would like to highlight that in our work from [21] we addressed for the first time the problem of scheduling the multicast flows on TSN networks.

Evaluation of Priority Assignment in TSN. To evaluate the GA-based strategy, the authors in [65] developed 3 other baseline strategies for assigning the priority. These strategies are **Fixed Priority Code Point** (**FPCP**), **Audsley's Optimal Priority Assignment** (**AOPA**) and **Robust Priority Assignment** (**RPA**). For assigning the number of express traffic queues, a single optimal ζ is used for all switches and ports. For evaluation, there are generated 400 synthetic test-cases. Each test-case consists of a tree-like topology and the traffic is generated such that the number of flows over classes of service are balanced.

As the results from [65] show, the load balance highly influences the schedulability of flows, therefore the GA and init strategies give the 100% schedulability even in case of ARQ. Moreover, the FPCP gives also good results in terms of schedulability, being able to obtain schedulable solutions in 320 and 400 out of 400 test-cases with ARQ and without ARQ, respectively. The reliability is drastically improved for solutions obtained with GA and init comparing with baseline strategies. Though, in terms of reliability, GA does not show such great improvement comparing with the best solution from the initial population. The results further show that the GA strategy with a single evolution is infeasible to be used for runtime (re)configuration, since the execution time increases exponentially with the number of flows.

5.4 Bandwidth Allocation

With the advent of asynchronous traffic shapers, the WCD of BS traffic can be upper bounded. Though, this upper limit can be calculated only if the switches are configured carefully and the BS traffic adheres to the bandwidth limitations. Therefore, the problem of bandwidth allocation is addressed in literature for AVB networks, in order to provide the values for the bandwidth limiting configuration parameters. These configuration parameters are called also operational parameters. For AVB, these operational parameters are represented by the idle slope, which indicates the percentage of the total speed that can be used by critical traffic.

The problem of bandwidth allocation is briefly addressed in [27] and [62] for AVB-based networks. The authors in [27] conduct first a series of simulations to compare the average end-to-end delay for AFDX, AVB/CBS and TSN/TAS networks. The aim of this paper is to determine the idle slopes such that the average and WCD is minimized for all critical flows. The main contribution

of the case-study in [62] is the evaluation against timeliness and throughput requirements of variations of IEEE 802.1Q strict priority, AVB/CBS and TSN/TAS protocols. The aim is to determine the protocol variations (and to some extent of configuration parameters) such that deadline and throughput constraints are satisfied while minimizing the implementation costs.

5.5 Scheduling

With the introduction of scheduled traffic in Ethernet-based networks, namely with advance of TTEthernet in 2011, the problem of determining the schedule tables gains a lot of interest from the research community. Let us recall that as input for the scheduling problem, there are the network topology $G(\mathcal{N}, \mathcal{E})$ and the set of flows $\mathcal{F}$, including their properties. As mentioned in Section 4, for the scheduling problem, the flows can be already assigned to routes. In brief, the scheduling problem means to determine the transmission time windows for all flows on each link. Due to its inherent complexity and different TS traffic models, there are proposed constructive solutions as well as optimization solutions that are attempting to improve the schedule tables regarding a mixture of aspects.

5.5.1 Scheduling for TSN-based Networks. The scheduling problem for TSN-based networks is also intensively addressed in literature. For example, in [13] are presented the constraints for obtaining the schedule tables for fully deterministic transmission of TS traffic. Let us recall that, comparing to TTEthernet, in TSN are scheduled the queues of egress ports. These per-queue schedule tables are defined by the so-called GCLs, which define when gates of associated queues are opened and closed.

Therefore, in TSN, TS flows that are sharing the same queue can experience non-deterministic transmissions due to the interleaving of frames of different flows. To overcome this non-deterministic transmission of TS traffic, the work in [13] extends the work in [80] with the flow and frame isolation constraints, respectively. We already presented these constraints in Section 4, when defining the constraints of the scheduling problem for TSN. To synthesize the GCLs, the authors in [13] use *the SMT and OMT solvers*.

About the same time, in [16] is addressed also the scheduling problem in TSN for a no-wait approach of transmitting the TS traffic. Besides computing schedule tables that minimize the end-to-end delay of TS traffic, the goal of the problem in [16] is to minimize the number of guard band events. Let us recall that the guard band mechanism closes the gates for non-scheduled traffic in advance to ensure that there are no lower priority frames on transmission when it is scheduled the transmission of TS traffic. Even if this is a necessary mechanism to ensure timeliness requirements for TS traffic, it is also a bandwidth wastage for non-scheduled traffic that has to be mitigated. To solve this scheduling problem, the authors in [16] are using multiple strategies, such as an *ILP solver*, a *list heuristic,* and a *TST*.

The no-wait transmission model for TS flows adopted in [16] means that the schedule tables are determined such that frames of TS traffic are not queued up on the switches. Namely, once the transmission of a frame from source has been started it is not interrupted by any other transmissions until that respective frame reaches its destination. Another interesting assumption is that all TS flows have the same period. Since solving the no-wait scheduling problem by mean of ILP solver is unfeasible for inputs with large number of flows, the authors in [16] propose an alternative solution. Namely, they use a metaheuristic approach to decide the order on which to schedule the flows and a heuristic approach to decide the starting time of transmitting each TS flow.

For deciding the order of scheduling the TS flows, in work from [16], it is used a tabu-search metaheuristic. To generate the initial solution, they are using five strategies, running thus the search strategy five times for each input. We would like to recall that the goal of work from [16] is to minimize the guard band events, therefore to schedule as many flows as possible back-to-back.

Therefore, in [16] to generate the neighboring solutions for a current solution, it is computed the schedule table. After computing the schedule, it is selected the flow that is reaching its destination at the latest point in time. Such flow is called critical flow. The neighboring set is obtained by (a) inserting before and (b) swapping the critical flow with all other flows that are preceding the critical flow on the current ordering. Furthermore, from the set of neighboring solutions, it is selected as next current solution the one (1) with the minimum receiving time for critical flow and (2) that is not in the tabu list. Alternatively, it can be selected as next current solution the neighboring one having a lower receiving time for the critical flow comparing to the best-so-far ordering solution. In the tabu list are stored the flows that were critical on the previous iterations. The search is stopped after a fixed number of iterations.

The actual computation of schedule tables for work in [16] is done using a *GH*. This strategy assigns start transmission times to flows as-soon-as-possible, ensuring that the transmission constraints are satisfied. While the ordering strategy attempts to reduce the compressing of schedule tables by decreasing the maximum receiving time of a flow, the authors in [16] use also a second strategy to further compress the schedule table. This second strategy attempts to postpone the starting time of transmitting a flow such that on consequent links multiple flows are scheduled back-to-back.

Based on the constraints defined in [13], there is a plethora of consequent works to improve the computation of GCLs. For example, authors in [67] present an ILP formulation which in general is able to reduce the running time comparing to the OMT formulation from [13]. For reconfiguring the TS flows at runtime, authors in [69] propose a heuristic to compute the GCLs for newly admitted TS flows. Here, it is used the *list scheduling heuristic* to determine the GCLs for a single TS flow on top of an already existing schedule table. The optimization of GCLs is addressed briefly in our previous work from [20]. Here, we are using a *GRASP* to find the GCLs of TS traffic for mixed-criticality applications in TSN.

Furthermore, the authors in [76] formulate the synthesis of GCLs using the array theory encoding. Here, the authors take into account also the limited number of gate entries that can be defined for an egress port. Therefore, giving the limited number of gate events, the time windows lengths are determined along with the assignment of frames to time windows such that GCLs constraints defined in [13] are satisfied. Introducing a new variable, namely the array of time windows, in [76] are adapted the constraints from [13] to take into account the new variable.

Moreover, the flow isolation constraint from [13] is extended in [76]. Now, the frames of 2 TS flows that are sharing an egress port of a switch and come from different ingress ports must: (1) be separated in the time domain, (2) use different queues, or (3) use the same queue and time window. Adding this third alternative to the flow isolation constraint, the work in [76] extends the search space of valid GCLs.

Evaluation of Scheduling in TSN. To evaluate their strategy, the authors in [13] have three sets of experiments. First, they study the feasibility of the strategy that uses the frame isolation constraint, second, for comparing the performance when using frame and flow isolation constraints and the third one is to study also the number of required queues for TS traffic. For these experiments there are used rather small topologies, with networks varying from 4 to 12 nodes. Furthermore, the number of flows is increased for each type of topology to preserve the link load. The performance of the strategy is studied also for three different set of priorities.

As the first set of experiments in [13] confirm, the runtime increases exponentially with the input. From the simplest to the most complex input combinations, the runtime variates between 1 second and 4 hours. Furthermore, as expected for the second set of experiments, by using the flow isolation reduces the runtime, in average with 13%, comparing to usage of frame isolation.

Surprisingly, for the most complex topology, with 10 flows and second and third set of periods, the strategy using the flow isolation does not reduce the runtime. From here we can conclude that deciding between usage of flow isolation and frame isolation, besides schedulability performance, makes sense only for loaded networks.

Furthermore, the third set of experiments from [13] show how flow and frame approaches behave also in terms of minimizing the number of queues used for TS traffic. We would like to recall, that depending on the configurations, it is possible at an egress port to use multiple queues for the same type of traffic. Therefore, with this set of experiments, the authors in [13] show which is the minimum number of queues used for TS traffic such that the transmission of TS flows is fully deterministic. Regarding number of queues, as the results show, the approach using the frame isolation improves better, for all cases using maximum 2 TS queues. Furthermore, we can observe that indeed, looking for optimized solutions takes longer than simply finding a satisfiable solution, the exception being for test-cases 17 and 20.

For the evaluation of strategy in [76], the authors generate multiple synthetic test-cases. They are using rather lab-sized test-cases, with networks up to 50 ESes and 10 switches and with up to 750 flows that have harmonic periods. In the first set of experiments, there is evaluated the runtime of the strategy with regard to number of flows and of used time windows. As the results show, in this context the metric that influences the most the running time is the number of time windows. It seems that the algorithm scales well for relatively small number of time windows, maximum 2 for example. But as reported, beyond that limit the runtime jumps from several minutes to hours, exceeding the time limit of 40 hours for 50 flows and medium-size topology.

The second set of experiments from [76] is studying the relation between number of time windows, jitter, and strategy runtime. As it can be observed, when using multiple time windows per queues of egress ports, the jitter is decreased. As expected, on the experiments studying the jitter, the running time also increases with the number of time windows, but in this case the increase is not that steep. An interesting aspect is that, for one of the optimization strategies, the running time even decreases when using 8-time windows comparing to case when using 7 windows.

In the end, the third set of experiments from [76] is studying the performance of the time windows-based solution comparing to the frame-based solution presented in [13]. As we can observe windows-based strategy behaves better comparing to frame-based strategy only for cases when using 1- and 2-time windows. For the basic experiments with 3-time windows experiments started to exceed the time out of 10 minutes already when using the 100 flows. Furthermore, to show the effectiveness of their method with the jitter optimization, the authors in [76] repeat the comparison with frame-based strategy for much smaller but with much higher bandwidth usage. In this case, they highlight that the jitter-based optimization solution is able to schedule several flows more in contrast with frame-based solution.

Finally, to evaluate their strategies, the authors in [16] use 30 test-cases, with networks variation between 29 and 120 nodes and with a number of flows between 30 and 1500. Their strategies are evaluated regarding three aspects: (1) improvement of flowspan, (2) scalability, and (3) the reduction of number of gate events. To study the flowspan improvement, the results obtained with tabu-search based strategy is compared with the results obtained with the ILP formulation run with a time limit of 5 hours. As the results show, the tabu-search strategies reduce (or have the same) flowspan for about 21 test-cases. In average, these strategies reduce the flowspan with about 3% comparing with the timed-out ILP strategy.

In terms of runtime, the tabu-search strategy presented in [16] gives rather good results, the runtime increasing polynomial regarding the number of flows. As we can observe from the results, for the 1,500 flows the schedule, using the tabu-search strategy, is obtained in a little bit more than 3 hours. Moreover, as the authors are reporting, by using the no-wait transmission scheme,

the computation of schedule is not influenced by the size of the topology. Though, we would be interested to see how the link load influence the runtime of this scheduling strategy. In the end, it is evaluated the effectiveness of compressing strategy. As the results show the compression has no effect in more than 50% of the test-cases (here being used about 75 test-cases). Though, when the compression strategy takes effect, it reduces the number of gates opening events with 24% in average.

5.6 Summary

As we could observe, the scheduling problem for TSN-based networks gained a tremendous interest last few years. This interest can be justified by the high complexity of the problem and the timeliness requirements that it can solve. A more detailed overview of the strategies that are used in solving this scheduling problem are presented in our previous work [23]. Therefore, throughout this section we did not insist on solutions for the scheduling problem, but on the solutions of the design problems that gave the go-ahead for the later problem addressed mainly for the TSN-based networks.

The Table 3 sums up the design strategies presented through this section. For each article are presented the addressed design problems (second column), the used type of strategy (third column), the targeted objectives (fourth column), and the fifth column presents the benefits of each article. The papers marked with "*" are addressing the non-scheduled traffic in TSN. In the table there are used some new abbreviations, namely Opt (for optimization strategy) and Const (for constructive strategy). Please note that, the KSP strategy is designated as "Const**", since multiple pathing solutions are constructed with this strategy, but the strategy by itself does not select a path.

6 DISCUSSIONS AND FUTURE DIRECTIONS

As several solutions for the networking design problems were presented in Section 5, our goal in this section is to discuss and relate the obtained results and to draw relations among strategy types. After discussing and identifying the limitations that the analyzed solutions have, we aim to suggest some future directions.

6.1 Discussed Solutions

As already introduced, Figure 4 shows, using a Venn diagram, an overview of the strategies used to solve different types of problems. We will use throughout this section the concept of strategy complexity. As we can see from the figure, there are multiple types of strategies that can solve the same problem. Therefore, we would like to highlight that the "strategy complexity" is different from the algorithmic complexity of a problem. There are multiple ways to assess the complexity of an application, but, because this is the metric that we could find in most of the analyzed papers, we will use the running time to evaluate the complexity of a strategy.

For example, as we can observe in 4 , the "**Greedy Heuristic**" (**GH**) is common for each type of problem; a different flavour of this strategy solving a specific design problem. For example, the GH for the topology selection problem is to construct the fully connected network as presented in our previous work from [22] addressed for non-scheduled traffic in TSN. As we could see from the presented solutions, the routing problem has the most variances of the GH, namely the SP, the load-adaptive and redundancy-aware heuristic and ECMP. The GH for priority and traffic type assignment problem is to assign the highest priority to flows with more strict deadlines, while for scheduling problem is the list heuristic.

The load-based and redundancy-driven routing heuristic from [64] is able to find a solution: (a) for 1,000 flows, in about 25 s, (b) for 25 nodes—rather small network, in less than a second and (c) for 120 nodes, in about 520 s. The scheduling heuristic as presented in [69] renders better

Table 3. Summary of Solutions for Designing TSN-Aware Networks that Accommodate Time-critical Applications; Each Solution (Paper Reference in Col. 1) Presented in Terms of: Addressed Design Problem (Col. 2), Used Type of Strategy (Col. 3), Objectives (Col. 4), and Benefits (Col. 5)

Ref.	Design problems	Strategy type	Objectives	Benefits
[22]*	Topology selection + Routing	Shortest-path (Const.) + GRASP (Opt.) + CP solver (Opt.)	minimize the topology cost For fault-tolerant applications	Formalizes constraints for multi-cast routing
[6]	Topology selection + Routing + Scheduling	KSP (Const.**) + List scheduling (Const.) + LP (Opt.)	Minimize topology cost For fault-tolerant applications	It helps to consider schedulability when determining the topology
[57]	Routing	KSP (Const.**) + GRASP (Opt.)	Minimize no. links used by BS flows + Minimize overall WCD of BS traffic	
[63]	Routing	ILP solver (Opt.)	Maximize time available to transmit non-TS traffic	Uses the no-wait transmission model
[64]	Routing	GH (Const.) + TST (Opt.)	Minimize maximum bandwidth used on a link	Considers fault-tolerant applications
[79]*	Priority assignment	SMT solver (Const.)	N.A.	
[65]	Priority and traffic type assignment	GA (Opt.)	Minimize overall WCD + Maximize no. retransmissions	Uses an interesting 2-genes individual
[21]	Traffic type assignment	List scheduling (Const.) + Tabu search (Opt.)	Maximize overall utility of soft real-time traffic	Introduces scheduling method for multi-cast traffic
[13]	Scheduling	SMT solver (Const.) OMT solver (Opt.)	Minimize queues used by TS traffic	Adapts the constraints to obtain fully deterministic transmission in TSN
[67]	Scheduling	ILP solver (Opt.)	Minimize running time	
[69]	Scheduling	List scheduling (Const.)	Minimize running time	Provides an online incremental GCLs synthesis
[20]	Scheduling	GRASP (Opt.)	Maximize no. schedulable BS flows	Optimizes GCLs for mixed-criticality applications
[76]	Scheduling	SMT solver (Const.)	Minimize no time windows	Considers upper bound on no. entries of GCL
[16]	Scheduling	ILP solver (Opt.) + List scheduling (Const.) + Tabu search (Opt.)	Minimize overall WCD of TS traffic + Maximize time available For non-TS traffic	Considers a no-wait transmission model

results in terms of runtime; obtaining a solution for a medium sized network (of 76 nodes) in about 3 seconds and for a large sized network (of 402 nodes) in about 60 seconds.

As stated in Section 5, the heuristic routing strategy from [64] is not suitable for online configuration, since the running time is rather long, namely of about 50 seconds for a network with 80 nodes. We expect that ECMP is a suitable routing heuristic to be used for online network configuration. But as the runtime for ECMP is not presented in any of the analyzed papers, we cannot give numeric results to proof this statement.

Another constructive strategy, that is not using heuristics though, is the **Satisfiability-Modulo Theory** (**SMT**). In [13] are shown the running times for SMT strategies that are solving the scheduling problem in TSN with two different types of constraints, namely frame isolation and flow isolation constraints. As the results show, for the moderately large test-cases, the SMT strategies are not suitable for online configuration since the strategy using the frame isolation constraint runs more than 300 seconds to find a solution while the strategy using the flow isolation constraint, the fastest of the two strategies, spends more than 15 seconds to find a scheduling solution.

The constructive strategies are fast, particularly the heuristics, being able to find good quality solutions, as demonstrated in the literature. Though, the constructive strategies do not guarantee neither that they can find a solution nor the optimality. On the other front, the CP, ILP, and OMT

methods are guaranteed to find the optimal solution when they can finish in reasonable time—this type of strategies being very slow, the runtime increasing exponentially with the system size. Besides size of the input, finding a *good* quality solution takes longer. For example, as highlighted by authors in [13] and [63], the runtime of SMT, OMT, and ILP strategies increases by adding more constraints.

A trade-off between heuristic and exhaustive strategies is represented by metaheuristics such as GA, GRASP, LP, and TST. The metaheuristics start with a set of initial solutions and explore the search space by generating neighboring solutions of a set of candidates. So, metaheuristics have the advantages of speed (can be used heuristics for initial solutions) and of obtaining *better* quality solutions (by evaluating more solutions). A well-known technique to prove that a metaheuristic gives good quality solutions is by comparing it with exhaustive search that is run for small examples. This technique is used by authors in [57] and [22] for example.

6.2 Future Directions

As we could see throughout this article, there exist strategies that provide good quality solutions for the problems of designing time-aware networks. There is always a trade-off between the computational resources that are available and the quality of the solutions that we can obtain. Summing up the previous discussion, the limitations of the analyzed design strategies are related to implementation costs of such strategies. Therefore, the identified limitations are about the most appropriate (a) architectures of the NC, as defined in the introductory section, and (b) online configuration. For each limitation, after presenting more details and intuitive solutions, we are going to give also some future directions towards the end of this section.

The applications for designing networks that can accommodate time-critical distributed applications rely on C&M planes as discussed previously in Section 1. Regarding availability of computational resources, traditionally in AVB, the network control was distributed, which means that each switch decides locally if to forward a BS flow or not. With the advances brought by TSN, the distributed design model is not enough anymore since some configuration parameters should be synchronized across the network. Therefore, TSN introduces also two centralized configuration models that are specified in iEEE 802.1Qcc amendment [45]. In this work, we are interested generically in a centralized approach, and, for more details about the differences between centralized models in IEEE 802.1Qcc, the reader is redirected to [45] and [68].

In a centralized approach, the NC can be implemented on different types of architectures. For example, the NC can be hosted on a server or it can reside on a cloud. Furthermore, a hosting server can be a dedicated HW or a general-purpose computer. The choice of the NC architecture highly depends on the usage area. For example, due to the limited space, we cannot deploy an in-vehicle (automotive) or onboard (airspace) NC on a cloud. But for controlling a smart city, medical devices from a hospital or the production floor of a smart-factory, the NC can be implemented over a dedicated cloud.

An intuitive mapping of types of strategies to NC architectures is as following. The GH strategies, on first hand, can be implemented on any types of architectures since this type of strategies is not that complex. On the other hand, the optimization strategies, both exhaustive and metaheuristics, are more suitable to be implemented on a server and cloud due to the fact that they are more computationally intense. As we highlighted before, the complexity of optimization strategies increases with the number of constraints. But if we implement the optimization strategies on a multi-core server or cloud, we can exploit an advantage of constraints-based strategies, namely we can run the search in parallel for independent sub-sets of the input.

In terms of online design strategies for time-aware networks, particularly regarding the traffic load, we consider that the following questions should be addressed in the future: (1) how should

look a configuration where new flows can be integrated and (2) how large is the dynamicity of time-aware networks. The network dynamicity means that the network topology and traffic can change at runtime and configuration parameters should be recomputed to accommodate the modification. Thus, the later question is further divided into: (i) how fast should be obtained the configuration of a newly integrated flow and (ii) how often are occurring the requests to integrate a new flow.

For question (1), the straight-forward answer, regarding the scheduling problem, is to compute schedule tables with high degree of porosity. A high-porosity schedule table is one where empty spaces are introduced between scheduled slots for facilitating the transmission of non-scheduled traffic. For example, such scheduling strategies are proposed in [81] for TTEthernet-based networks and authors in [61] discuss about the impact of porosity of schedule tables over the performance of BS traffic for TSN-based networks. Regarding question (2), to the best of our knowledge, there are no answers provided yet in the literature. In general, the approach is to obtain the design of time-critical network by extending the solutions that exist for real-time systems. Therefore, we suggest that a good direction for answering the questions about time-aware networks dynamicity, again for scheduling problem, is to adapt the scheduling strategy of elastic applications as presented in [74].

As we could see, the presented solutions are theoretical and have their advantages. To see though these theoretical results at work, there are more steps needed, such as defining the requirements and strategies for online design of time-critical networks. From our side, we would like in the future to study in more details the most suitable NC architecture for each area. Namely, we would like to investigate which is the optimal architecture depending on: (1) the traffic requirements, (2) network sizes, and (3) physical environment of an area.

7 CONCLUSION

This article brings a detailed overview of design strategies for networks that are supporting time-critical applications. The article also presents the improvements brought by each protocol and the challenges that should be further addressed, particularly for TSN, before TSN's benefits can be fully exploited. TSN is a suite of amendments and standards, and it has the potential to reduce the cost of network deployment while satisfying the reliability and timeliness requirements of time-critical applications. Before year 2000, such requirements could be met only by using customized proprietary solutions, but now the costs can be reduced by using standardized solutions. To satisfy though the reliability and timeliness requirements, the network should be designed appropriately.

A broad introduction is devoted to evolution of Ethernet-based time-aware communication protocols, i.e., evolution of transmission mechanisms that lead to enhancements offered today by TSN. In this article, efforts are spent also on formalizing the problems (design tasks) encountered on designing networks for time-critical applications. The defined design tasks are: (1) network planning and design, (2) routing, (3) priority and traffic type assignment, (4) bandwidth allocation, and (5) scheduling. Furthermore, the work towards solving such design problems has been presented. In this article, there are investigated strategies where critical aspects such as the need for (1) increasing reliability, (2) improving timeliness, (3) growing network throughput, (4) reducing network cost, and (5) decreasing runtime computation of network configuration. First, constructive strategies for designing time-aware networks have been presented, followed by optimization strategies.

While the Ethernet-based communication protocols available for time-critical applications are clearly defined, more research is needed on design strategies that can find optimal network configuration. By optimal network configuration we understand a configuration that maximizes the benefits behind such time-aware protocols and minimizes costs of network deployment. Businesses become interested in time-aware (particularly TSN) solutions due to the fact that they offer potential improvements on reliability and timeliness through COTS components. Meanwhile, TSN solutions

are suitable for multiple timing and safety aware areas, such as (but not limited to) avionics, automotive, industrial automation, and fronthaul mobile networks. The adoption of Ethernet-based time-aware protocols, such as TSN, is justified by businesses taking into account the alternative standardized network components with their innate reduced non-recurring engineering and configuration costs.

ACKNOWLEDGMENTS

To Zifan Zhou for drawing some of the figures.

REFERENCES

[1] 2019. 5G Wireless Fronthaul Requirements in a Passive Optical Network Context. (2019). Retrieved May 2021 from https://www.itu.int/ITU-T/recommendations/rec.aspx?rec=13991&lang=en.
[2] 2021. Release 17 Update from SA2. (2021). Retrieved August 2021 from https://www.3gpp.org/news-events/2200-sa2_artcle.
[3] Aeronautical Radio, Inc. 2005. ARINC 664P7: Aircraft Data Network, Part 7, Avionics Full-DupleX Switched Ethernet Network (AFDX). (2005). Retrieved April 2020 from https://global.ihs.com/doc_detail.cfm?&item_s_key=00465045&item_key_date=940604&input_doc_number=&input_doc_title=&origin=HISC.
[4] Aeronautical Radio, Inc. 2009. ARINC 664P7: Aircraft Data Network, Part 7, Avionics Full-DupleX Switched Ethernet Network (AFDX). (2009). Retrieved April 2020 from https://standards.globalspec.com/std/1283307/ARINC%20664%20P7.
[5] Ahmad Al Sheikh, Olivier Brun, Maxime Chéramy, and Pierre-Emmanuel Hladik. 2013. Optimal design of virtual links in AFDX networks. *Real-Time Systems* 49, 3 (2013), 308–336.
[6] Ayman A. Atallah, Ghaith Bany Hamad, and Otmane Ait Mohamed. 2018. Fault-resilient topology planning and traffic configuration for IEEE 802.1Qbv TSN networks. In *Proceedings of the IEEE International Symposium on On-Line Testing And Robust System Design*. 151–156.
[7] Henri Bauer, Jean Luc Scharbarg, and Christian Fraboul. 2010. Improving the worst-case delay analysis of an AFDX network using an optimized trajectory approach. *IEEE Transactions on Industrial Informatics* 6, 4 (2010), 521–533.
[8] Henri Bauer, Jean-Luc Scharbarg, and Christian Fraboul. 2010. Worst-case end-to-end delay analysis of an avionics AFDX network. In *Proceedings of the Conference and Exhibition on Design, Automation Test in Europe*. 1220–1224.
[9] Henri Bauer, Jean-Luc Scharbarg, and Christian Fraboul. 2012. Applying trajectory approach with static priority queuing for improving the use of available afdx resources. *Real-Time Systems* 48, 1 (2012), 101–133.
[10] Edmund K. Burke and Graham (Eds.) Kendall. 2014. *Search Methodologies*. Springer. 716.
[11] Giorgio Buttazzo, Giuseppe Lipari, Luca Abeni, and Marco Caccamo. 2005. *Soft Real-Time Systems*. Springer. 283
[12] FlexRay Consortium. 2006. FlexRay. Retrieved on April 2020 from https://piembsystech.com/flexray-protocol/.
[13] Silviu S. Craciunas, Ramon Serna Oliver, Martin Chmelík, and Wilfried Steiner. 2016. Scheduling real-time communication in IEEE 802.1Qbv time sensitive networks. In *Proceedings of the International Conference on Real-Time Networks and Systems*. 183–192.
[14] Ruiz De Azua, Joan Adria, and Boyer Marc. 2014. Complete modelling of AVB in network calculus framework. In *Proceedings of the International Conference on Real-Time Networks and Systems*. 55–64.
[15] Jean-Dominique Decotignie. 2005. Ethernet-based real-time and industrial communications. *Proceedings of the IEEE* 93, 6 (2005), 1102–1117.
[16] Frank Duerr and Naresh Ganesh Nayak. 2016. No-wait packet scheduling for IEEE time-sensitive networks (TSN). In *Proceedings of the International Conference on Real-Time Networks and Systems*. 203–212.
[17] Jeff A. Estefan. 2007. Survey of model-based systems engineering (MBSE) methodologies. *INCOSE MBSE Focus Group* 25, 8 (2007), 1–12.
[18] EtherCAT Technology Group. 2003. EtherCAT (Ethernet for Control Automation Technology). https://www.ethercat.org/default.htm.
[19] Fabrice Frances, Christian Fraboul, and Jérôme Grieu. 2006. Using network calculus to optimize the AFDX network. In *Proceedings of the European Congress of Embedded Real-Time Software*. 8s.
[20] Voica Gavrilut and Paul Pop. 2018. Scheduling in time sensitive networks (TSN) for mixed-criticality industrial applications. In *Proceedings of the International Workshop on Factory Communication Systems*. 1–4.
[21] Voica Gavrilut and Paul Pop. 2020. Traffic-type assignment for TSN-based mixed-criticality cyber-physical systems. *ACM Transactions on Cyber-Physical Systems* 4, 2 (2020), 1–27.
[22] Voica Gavrilut, Bahram Zarrin, Paul Pop, and Soheil Samii. 2017. Fault-tolerant topology and routing synthesis for IEEE time-sensitive networking. In *Proceedings of the International Conference on Real-Time Networks and Systems*. 267–276.

[23] Voica Gavrilut, Luxi Zhao, Michael L. Raagaard, and Paul Pop. 2018 (in press). AVB-aware routing and scheduling of time-triggered traffic for TSN. *IEEE Access* 6, 1 (2018 (in press)), 14s.
[24] Abenezer Girma, Niloofar Bahadori, Mrinmoy Sarkar, Tadewos G. Tadewos, Mohammad R. Behnia, M. Nabil Mahmoud, Ali Karimoddini, and Abdollah Homaifar. 2020. IoT-enabled autonomous system collaboration for disaster-area management. *IEEE/CAA Journal of Automatica Sinica* 7, 5 (2020), 1249–1262.
[25] Ethan Grossman. 2019. Deterministic Networking Use Cases. *RFC* 8578, (2019), 1–97. Retrieved from https://datatracker.ietf.org/doc/html/rfc8578.
[26] Evangelos Haleplidis, Kostas Pentikousis, Spyros Denazis, Jamal Hadi Salim, David Meyer, and Odysseas Koufopavlou. 2015. Software-Defined Networking (SDN): Layers and Architecture Terminology. *RFC* 7426, (2015), 1–35. Retrieved from https://rfc-editor.org/rfc/rfc7426.txt.
[27] Feng He, Lin Zhao, and Ershuai Li. 2017. Impact analysis of flow shaping in ethernet-AVB/TSN and AFDX from network calculus and simulation perspective. *Sensors* 17, 5 (2017), 1181.
[28] Honeywell. 1992. SAFEbus. https://ieeexplore.ieee.org/document/282179.
[29] IEEE 802.1 Working Group. 2005. IEEE 802.1Q-2005 - 802.1Q Revision. (2005). Retrieved April 2020 from http://www.ieee802.org/1/pages/802.1Q-2005.html.
[30] IEEE 802.1 Working Group. 2009. IEEE 802.1Qav-2009 - IEEE Standard for Local and Metropolitan Area Networks–Virtual Bridged Local Area Networks Amendment 12: Forwarding and Queuing Enhancements for Time-Sensitive Streams. (2009). Retrieved April 2020 from https://standards.ieee.org/standard/802_1Qav-2009.html.
[31] IEEE 802.1 Working Group. 2010. IEEE 802.1Qat-2010 - IEEE Standard for Local and Metropolitan Area Networks - Virtual Bridged Local Area Networks - Amendment 14: Stream Reservation Protocol. (2010). Retrieved April 2020 from https://standards.ieee.org/standard/802_1Qat-2010.html.
[32] IEEE 802.1 Working Group. 2011. IEEE 802.1BA - Audio Video Bridging (AVB) Systems. (2011). Retrieved April 2020 from http://www.ieee802.org/1/pages/802.1ba.html.
[33] IEEE 802.1 Working Group. 2011. IEEE 802.1Q-2011 - IEEE Standard for Local and Metropolitan Area Networks. (2011). Retrieved April 2020 from https://standards.ieee.org/standard/802_1Q-2011.html.
[34] IEEE 802.1 Working Group. 2018. IEEE 802.1Q-2018 - IEEE Standard for Local and Metropolitan Area Networks. (2018). Retrieved April 2020 from https://standards.ieee.org/standard/802_1Q-2018.html.
[35] IEEE 802.3 Ethernet Working Group. 2016. IEEE 802.3br-2016 - IEEE Standard for Ethernet - Amendment 5: Specification and Management Parameters for Interspersing Express Traffic. (2016). Retrieved April 2020 from https://standards.ieee.org/standard/802_3br-2016.html.
[36] IEEE 802.3 Working Group. 2018. IEEE 802.3-2018 - IEEE Standard for Ethernet. (2018). Retrieved April 2020 from http://www.ieee802.org/3/cj/index.html.
[37] IEEE AVB Task Group. 2011. Audio-Video Bridging (AVB). (2011). Retrieved April 2020 from http://www.ieee802.org/1/pages/avbridges.html.
[38] IEEE TSN Task Group. 2012. Time-Sensitive Networking (TSN). https://1.ieee802.org/tsn/.
[39] IEEE TSN Task Group. 2015. IEEE 802.1Qbv-2015 - IEEE Standard for Local and Metropolitan Area Networks - Bridges and Bridged Networks - Amendment 25: Enhancements for Scheduled Traffic. (2015). Retrieved April 2020 from https://standards.ieee.org/standard/802_1Qbv-2015.html.
[40] IEEE TSN Task Group. 2015. IEEE 802.1Qca-2015 - IEEE Standard for Local and Metropolitan Area Networks- Bridges and Bridged Networks - Amendment 24: Path Control and Reservation. (2015). Retrieved April 2020 from https://standards.ieee.org/standard/802_1Qca-2015.html.
[41] IEEE TSN Task Group. 2016. IEEE 802.1Qbu-2016 - IEEE Standard for Local and Metropolitan Area Networks - Bridges and Bridged Networks - Amendment 26: Frame Preemption. (2016). Retrieved April 2020 from https://standards.ieee.org/standard/802_1Qbu-2016.html.
[42] IEEE TSN Task Group. 2017. IEEE 802.1CB-2017 - IEEE Standard for Local and Metropolitan Area Networks - Frame Replication and Elimination for Reliability. (2017). Retrieved April 2020 from https://standards.ieee.org/standard/802_1CB-2017.html.
[43] IEEE TSN Task Group. 2017. IEEE 802.1Qci-2017 - IEEE Standard for Local and Metropolitan Area Networks–Bridges and Bridged Networks–Amendment 28: Per-Stream Filtering and Policing. (2017). Retrieved April 2020 from https://standards.ieee.org/standard/802_1Qci-2017.html.
[44] IEEE TSN Task Group. 2018. IEEE 802.1CM - Time-Sensitive Networking for Fronthaul. (2018). Retrieved April 2020 from https://1.ieee802.org/tsn/802-1cm.
[45] IEEE TSN Task Group. 2018. IEEE 802.1Qcc-2018 - IEEE Standard for Local and Metropolitan Area Networks-Bridges and Bridged Networks - Amendment 31: Stream Reservation Protocol (SRP) Enhancements and Performance Improvements. (2018). Retrieved April 2020 from https://standards.ieee.org/standard/802_1Qcc-2018.html.
[46] IEEE TSN Task Group. 2020. IEC/IEEE 60802 - TSN Profile for Industrial Automation. (2020). Retrieved March 2021 from https://1.ieee802.org/tsn/iec-ieee-60802.

[47] IEEE TSN Task Group. 2020. IEEE 802.1AS-2020 - IEEE Approved Draft Standard for Local and Metropolitan Area Networks - Timing and Synchronization for Time-Sensitive Applications. (2020). Retrieved March 2021 from https://standards.ieee.org/standard/802_1AS-2020.html.
[48] IEEE TSN Task Group. 2020. IEEE 802.1DG - TSN Profile for Automotive In-Vehicle Ethernet Communications. (2020). Retrieved March 2021 from https://1.ieee802.org/tsn/802-1dg.
[49] IEEE TSN Task Group. 2021. IEEE 802.1DP - TSN for Aerospace Onboard Ethernet Communications. (2021). Retrieved August 2021 from https://1.ieee802.org/tsn/802-1dp.
[50] Profibus & Profinet International. 2001. ProfiNet. (2001).
[51] International Electrotechnical Commission (IEC). 1998. Fieldbus. Retrieved on April 2020 from https://www.fieldcommgroup.org/technologies/foundation-fieldbus.
[52] ISO Electrical and electronic components and general system aspects Technical Committee. 2018. ISO 26262:2018 - Road Vehicles – Functional Safety. (2018). Retrieved April 2020 from https://www.iso.org/standard/68383.html.
[53] Hadriel Kaplan and Robert Noseworthy. 2001. The Ethernet Evolution: From 10 Meg to 10 Gig: How it All Works! (2001). Retrieved April 2020 from https://www.iol.unh.edu/sites/default/files/knowledgebase/ethernet/ethernet_evolution.pdf.
[54] Hermann Kopetz. 1991. Event-triggered versus time-triggered real-time systems. *Lecture Notes in Computer Science* 563 (1991), 87–101.
[55] Hermann Kopetz. 2011. *Real-time Systems: Design Principles for Distributed Embedded Applications*. Springer. 376s.
[56] Hermann Kopetz and Guenther Gruensteidl. 1993. TTP - A time-triggered protocol for fault-tolerant real-time systems. In *Proceedings of the International Symposium on Fault-Tolerant Computing*. 524–533.
[57] Sune M. Laursen, Paul Pop, and Wilfried Steiner. 2017. Routing optimization of AVB streams in TSN networks. *ACM Sigbed Review* 13, 4 (2017), 43–48.
[58] Xing Liu, Ziyuan Cai, Huipu Fan, and Ming Yu. 2020. Experimental studies on the RTEthernet-based centralized fault management system for smart grids. *Electric Power Systems Research* 181, 106163 (2020), 106–163.
[59] Dorin Maxim and Ye-Qiong Song. 2017. Delay analysis of AVB traffic in Time-Sensitive Networks. In *Proceedings of the International Conference on Real-Time Networks and Systems*. 18–27.
[60] Robert M. Metcalfe, David R. Boggs, Charles P. Tacker, and Butler W. Lampson. 1975. Multipoint Data Communication System with Collision Detection. Patent No. 4,063,220, Issued Dec. 13th., 1977.
[61] Meyer, Philipp and Steinbach, Till and Korf, Franz and Schmidt, Thomas C. 2013. Extending IEEE 802.1 AVB with time-triggered scheduling: A simulation study of the coexistence of synchronous and asynchronous traffic. In *Proceedings of the IEEE Vehicular Networking Conference*. 47–54.
[62] Joern Migge, Josetxo Villanueva, Nicolas Navet, and Marc Boyer. 2018. Insights on the performance and configuration of AVB and TSN in automotive ethernet networks. In *Proceedings of the European Congress on Embedded Real Time Software and Systems*. 10s (Online only).
[63] Naresh G. Nayak, Frank Duerr, and Kurt Rothermel. 2018. Routing algorithms for IEEE802.1Qbv networks. *ACM Sigbed Review* 15, 3 (2018), 13–18.
[64] Mubarak Adetunji Ojewale and Patrick Meumeu Yomsi. 2020. Routing heuristics for load-balanced transmission in TSN-based networks. *ACM Sigbed Review* 16, 4 (2020), 20–25.
[65] Taeju Park, Soheil Samii, and Kang G. Shin. 2019. Design optimization of frame preemption in real-time switched ethernet. In *Proceedings of the Conference and Exhibition on Design, Automation Test in Europe*. 420–425.
[66] Paulo Pedreiras, Paolo Gai, Luis Almeida, and Giorgio Buttazzo. 2005. FTT-Ethernet: A flexible real-time communication protocol that supports dynamic QoS management on Ethernet-based systems. *IEEE Transactions on Industrial Informatics* 1, 3 (2005), 162–172.
[67] Paul Pop, Michael L. Raagaard, Silviu S. Craciunas, and Wilfried Steiner. 2016. Design optimization of cyber-physical distributed systems using IEEE time-sensitive networks (TSN). *IET Cyber-Physical Systems: Theory & Applications* 1, 1 (2016), 86–94.
[68] Aleksander Pruski, Mubarak Adetunji Ojewale, Voica Gavrilut, Patrick Meumeu Yomsi, Michael Stubert Berger, and Luis Almeida. 2021. Implementation cost comparison of TSN traffic control mechanisms. In *Proceedings of the IEEE Conference on Emerging Technologies and Factory Automation*. 1–8 (In Press).
[69] Michael L. Raagaard, Paul Pop, Marina Gutierrez, and Wilfried Steiner. 2017. Runtime reconfiguration of time-sensitive networking (TSN) schedules for Fog Computing. In *Proceedings of the IEEE Fog World Congress*. 1–6.
[70] Frederic Ridouard, Jean-Luc Scharbarg, and Christian Fraboul. 2008. Probabilistic upper bounds for heterogeneous flows using a static priority queueing on an AFDX network. In *Proceedings of the IEEE International Conference of Emerging Technologies and Factory Automation*. 1220–1227.
[71] Robert Bosch GmbH. 1986. Controller Area Network (CAN). https://www.bosch.com/stories/the-controller-area-network/.
[72] Obermaisser (Ed.) Roman. 2012. *Time-Triggered Communication*. CRC Press. 523s.

[73] SAE International. 2011. AS6802: Time-Triggered Ethernet. (2011). Retrieved April 2020 from https://www.sae.org/standards/content/as6802.
[74] Shaik Mohammed Salman, Saad Mubeen, Filip Markovic, Alessandro Papadopoulos, and Thomas Nolte. 2021. Scheduling elastic applications in compositional real-time systems. In *Proceedings of the IEEE Conference on Emerging Technologies and Factory Automation*. 1–8 (In Press).
[75] Rich Seifert and James Edwards. 2008. *The All-New Switch Book: The Complete Guide to LAN Switching Technology* (2nd ed ed.). Wiley Pub, Indianapolis, IN.
[76] Ramon Serna Oliver, Silviu S. Craciunas, and Wilfried Steiner. 2018. IEEE 802.1Qbv gate control list synthesis using array theory encoding. In *Proceedings of the Real-Time and Embedded Technology and Applications Symposium*. 13–24.
[77] Zhengguo Sheng, Saskia Pfersich, Alice Eldridge, Jianshan Zhou, Daxin Tian, and Victor C. M. Leung. 2019. Wireless acoustic sensor networks and edge computing for rapid acoustic monitoring. *IEEE/CAA Journal of Automatica Sinica* 6, 1 (2019), 64–74.
[78] Johannes Specht and Soheil Samii. 2016. Urgency-based scheduler for time-sensitive switched ethernet networks. In *Proceedings of the Euromicro Conference on Real-Time Systems*. 75–85.
[79] Johannes Specht and Soheil Samii. 2017. Synthesis of queue and priority assignment for asynchronous traffic shaping in switched ethernet. In *Proceedings of the IEEE Real-Time Systems Symposium*. 178–187.
[80] Wilfried Steiner. 2010. An evaluation of SMT-based schedule synthesis for time-triggered multi-hop networks. In *Proceedings of the Real-Time Systems Symposium*. 375–384.
[81] Wilfried Steiner. 2011. Synthesis of static communication schedules for Mixed-criticality systems. In *Proceedings of the IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops*. 11–18.
[82] Domitian Tamas-Selicean, Paul Pop, and Wilfried Steiner. 2015. Timing analysis of rate constrained traffic for the TTEthernet communication protocol. In *Proceedings of the IEEE International Symposium on Real-Time Distributed Computing*. 119–126.
[83] Andrew Tanenbaum and David Wetherall. 2011. *Computer Networks*. Prentice-Hall,. 938s
[84] United States Department of Transportation (DOT). 1988. Advisory Circular (AC) 25.1309-1A - System Design and Analysis. (1988). Retrieved April 2020 from https://www.faa.gov/regulations_policies/advisory_circulars/index.cfm/go/document.information/documentID/22680.
[85] Luxi Zhao, Paul Pop, and Silviu Craciunas. 2018. Worst-case latency analysis for IEEE 802.1 Qbv time sensitive networks using network calculus. *IEEE Access* 6, 1 (2018), 41803–41815.
[86] Luxi Zhao, Paul Pop, Qiao Li, Junyan Chen, and Huagang Xiong. 2017. Timing analysis of rate-constrained traffic in TTEthernet using network calculus. *Real-Time Systems* 53, 2 (2017), 254–287.
[87] Luxi Zhao, Paul Pop, Zhong Zheng, and Qiao Li. 2018. Timing analysis of AVB traffic in TSN networks using network calculus. In *Proceedings of the IEEE Real-Time and Embedded Technology and Applications Symposium*. 25–36.
[88] Zifan Zhou, Juho Lee, Michael Stübert Berger, Sungkwon Park, and Ying Yan. 2021. Simulating TSN traffic scheduling and shaping for future automotive ethernet. *Journal of Communications and Networks* 23, 1 (2021), 53–62.

Received March 2021; revised September 2021; accepted November 2021