electronics

MDPI

Article

Software-Defined Time-Sensitive Networking for Cross-Domain Deterministic Transmission

Mengjie Guo, Guochu Shou *, Yaqiong Liu and Yihong Hu

School of Information and Communication Engineering, Beijing University of Posts and Telecommunications, Beijing 100876, China; mjguo@bupt.edu.cn (M.G.); liuyaqiong@bupt.edu.cn (Y.L.); yhhu@bupt.edu.cn (Y.H.)
* Correspondence: gcshou@bupt.edu.cn

Abstract: With the rapid development of Industrial 4.0, massive emerging time-critical applications pose new demands for industrial networks, such as strict latency boundaries, ultra-reliable transmission, and so on. Time-Sensitive Networking (TSN) is well suited for these demanding scenarios due to its support for deterministic transmission based on Ethernet. However, many use cases in real industrial scenarios involve non-TSN domains, and, thus, the network requires providing deterministic transmission across non-TSN domains to support them. To this end, this paper proposes a novel Software-Defined Time-Sensitive Networking (SD-TSN) framework that integrates the determinism of TSN and scalability of Software-Defined Networking (SDN). SD-TSN exploits the coordination between the coordinated controller and different domain controllers to bound non-deterministic queuing delays in a way that guarantees determinism. In particular, we designed a multi-domain time-aware traffic scheduling model, which harmonizes the time-aware shaper (TAS) in TSN and time-slots in non-TSN domains based on a global view, generating transmission schedules for each domain. A prototype testbed was built to conduct the experiments. The evaluation results demonstrate that the proposed method can efficiently achieve bounded delay and low delay variations in the transmission across non-TSN domains.

Keywords: software-defined time-sensitive networking (SD-TSN); multiple-domain traffic scheduling; deterministic transmission

Citation: Guo, M.; Shou, G.; Liu, Y.; Hu, Y. Software-Defined Time-Sensitive Networking for Cross-Domain Deterministic Transmission. *Electronics* **2024**, *13*, 1246. https://doi.org/10.3390/electronics13071246

Academic Editor: Ivan Ganchev

Received: 14 February 2024
Revised: 12 March 2024
Accepted: 23 March 2024
Published: 27 March 2024

1. Introduction

With the rapid development of Industry 4.0, the number of time-sensitive applications is increasingly growing [1]. These applications pose new requirements for the network in terms of determinism, i.e., the data need to be transmitted with a bounded delay and low delay variation [2,3]. Ethernet is popular in various fields in terms of scalability and cost. However, satisfying these demanding requirements over standard Ethernet is hard due to the best-effort (BE) transmission mechanism [4].

Time-Sensitive Networking (TSN) is a promising technology that provides deterministic, real-time, and ultra-reliable transmission through IEEE 802 networks [5]. Within TSN, the IEEE 802.1Qbv time-aware shaper (TAS) introduces a transmission gate per queue in front of the egress port to activate or deactivate the queue [6]. The gate is in the open state to activate the queue such that the queued frames in the queue are permitted to be transmitted, and in the case where the gate is in a closed state, the frames of this queue are not transmitted [7]. The transmission order of the frames in the signal queue depends on the arrival order, and the transmission selection among multiple open queues is a strict priority algorithm based on priority [8]. Furthermore, the state of all gates is controlled by the gate control list (GCL). Overall, all frames are assigned to the corresponding queues based on the priority code point in their headers, then GCL determines the transmission behavior of the frames such that it guarantees determinism. Although TSN is a rather novel technology, it is being widely adopted in a closed network to establish deterministic connections, e.g., in-vehicle networks and industrial automation [9].

However, the end-to-end transmission of many applications in real scenarios tends to involve non-TSN domains. For instance, within a factory or industrial park, diverse devices (e.g., sensors, actuators, machines, and robots) will continuously interact with each other through the network, and they require deterministic transmission guarantees to facilitate intelligent production and service with data and information [10]. TSN struggles to provide determinism for these applications due to the benefits of it typically being limited to closed networks [9]. First, it would be impractical to enable all equipment to support hardware TSN mechanisms for achieving large-scale deterministic transmission since replacing all legacy devices with TSN-capable equipment is highly expensive. On the other hand, under the wave of Information Technology (IT) and Operational Technology (OT) convergence, various terminals (e.g., Human–Machine Interface and laptops) from different vendors cannot fully support TSN capabilities. Moreover, even enabling all devices to support hardware TSN mechanisms regardless of cost, guaranteeing determinism in a large-scale network through TSN, is still challenging due to the strict requirements for time synchronization and short link distances [11]. Therefore, the need to extend TSN's capability to provide deterministic transmission across non-TSN domains is urgent.

Software-Defined Networking (SDN) is an attractive networking paradigm that separates the network control logic from the data plane, promoting the ability to program the network and facilitating network evolution [12]. After years of development and evolution, SDN has become quite mature and is being used in various fields, e.g., industrial [13], vehicle [14], and 5G [15]. Moreover, the fully centralized configuration model of TSN is highly fitting for the centralized control paradigm of SDN. Hence, this study combines the SDN paradigm and the fully centralized model of TSN, proposing a novel Software-Defined Time-Sensitive Networking (SD-TSN) framework to provide deterministic cross-domain transmission. Within SD-TSN, each domain controller is responsible for the network policy of its respective network, and the coordinated controller is responsible for the communication among them to establish the cross-domain transmission. SD-TSN exploits the global view to coordinate the domain controllers and the coordinated controller to bound queuing delays of time-sensitive traffic flows in non-TSN domains in a way that provides deterministic guarantees. In particular, we present a multi-domain time-aware scheduling model that generates transmission schedules for time-sensitive traffic flows in each domain by harmonizing the TAS in TSN domains and time slots in non-TSN domains. In general, the main contributions of this paper are as follows:

(1) We propose an SD-TSN framework that builds end-to-end transmission across non-TSN domains for time-sensitive traffic flows. It exploits the coordination between the coordinated controller and different domain controllers to bound queuing delays in non-TSN domains to provide deterministic guarantees.
(2) Within SD-TSN, we design a multi-domain, time-aware traffic scheduling model that harmonizes the TAS in TSN domains and time slots in non-TSN domains based on the global view to generate transmission schedules for time-sensitive traffic flows in each domain.
(3) We conducted extensive experiments in the simulation environment and prototype testbed. The experimental results demonstrate that the proposed method can provide bounded delay and low delay variation guarantees for transmission across non-TSN domains.

This paper is organized as follows. Section 2 presents the related work. Section 3 describes the design of the SD-TSN framework. Section 4 presents the multi-domain time-aware traffic scheduling model. Section 5 evaluates the performance of our proposed model and algorithm. Finally, Section 6 concludes this paper and presents future work.

2. Related Work

With the rise of TSN, extensive studies on deterministic transmission have been published [16–24]. The approaches in these studies are feasible but typically oriented to TSN networks and thus cannot provide deterministic transmission across non-TSN

domains. For this issue, some works investigated the extension of TSN through various advanced technologies to provide determinism. In general, they can be divided into two categories: wired-based solutions and wireless-based solutions.

Wire-based solution: The IETF deterministic networking (DetNet) working group defines an overall architecture to support transmission in layer three with deterministic latency, jitter, and very low/zero packet loss. [9]. Therefore, Peng et al. [25] designed a large-scale deterministic network flow-shaping mechanism to guarantee deterministic worst-case latency and zero packet loss for time-sensitive flows in Large-scale Deterministic Networks (LDNs). Krolikowski et al. [26] formulated the joint routing and scheduling problem for DetNet and provide efficient algorithms in large-scale deterministic networks. DIP is a scalable three-layer deterministic network that has the potential to become an LDN architecture due to its support for remote deterministic transmission services and substantial traffic management [27]. Tan et al. [11] proposed a hierarchical network containing access networks and a core network, which leverages Deterministic IP (DIP) in the core network to connect TSN networks for achieving long-distance deterministic transmission. Huang et al. [28] proposed a C-TSDN architecture that combines source routing, network slicing, and cycle-based scheduling technologies to ensure end-to-end deterministic transmission. Moreover, the integration of TSN and SDN is a promising solution that exploits centralized control to build the interoperability between TSN networks [9]. For instance, Xue et al. [7] propose an interconnection scheme following a software-defined TSN (SD-TSN) paradigm to enable deterministic communications among multiple closed networks. Zhong et al. [29] proposed an SDN-enable optical transport network simulation platform to support the end-to-end service resource prejudge in cross-domain TSN.

Wireless-based solution: This part describes the related work regarding extending TSN over wireless networks. First, given that both 802.11 and 802.3 are based on the same family of IEEE 802 standards [30], extending TSN over 802.11 is a natural and seamless step [31]. For instance, Seijo et al. [32] discuss the integration challenges of wired TSN and wireless local area network technologies and propose a hybrid TSN device architecture. Miranda et al. [33] present a modular, multi-domain controller architecture to provide end-to-end TSN-enabled control over LAN and WLAN domains. Cavalcanti et al. [31] describe the approaches to extend TSN to wireless with IEEE 802.11, containing the wireless network management model, wireless time synchronization, time-aware scheduling, wireless link reliability, etc. Furthermore, extending TSN through 5G is considered one of the best options to support Industry 4.0 as it supports ultra-reliable and ultra-low latency communications (URLLC). Wang et al. [34] proposed an integrated TSN and 5G industrial network architecture, where the 5G system acts as a logical TSN-capable bridge support deterministic communication. Atiq et al. [35] analyzed the recent standardization efforts and developments in IEEE 802.11 and 5G to enable low-latency, deterministic communications and present the current status of their integration with wired TSN. Larrañaga et al. [36] discuss current research and standardization work on 5G-TSN integration and quantify the 5GS bridge delay for a closed loop control application.

The above literature shows the research progress on deterministic transmission across non-TSN domains, summarized in Table 1. They mainly focus on extending TSN in the data plane with advanced technologies, which may be difficult and introduce extra costs. For example, the presented hierarchical architecture in [11] requires DIP to connect different TSN networks such that it enables long-distance deterministic transmission. In addition, 5G-based solutions typically need the device-side TSN translator (DS-TT) and network-side TSN translator (NW-TT) on the input and output ports of the 5GS, respectively, to enable process and forward data like Ethernet ports. This increases the cost of realistic deployment, and the transmission performance is easily restricted by the capabilities of these TSN translators. Unlike the above related works, the proposed SD-TSN focuses on extending TSN through the control-plane capabilities without imposing demanding requirements on the data plane. Depending on application requirements, environmental constraints and other factors, it can be wire-based, wireless-based, or a hybrid of both. Within SD-TSN,

based on the global view and centralized control capabilities, the non-TSN domain acts as a logic bridge whose delay can be measured. Incorporated with the advanced tools developed for TSN, the controller will specify schedules for different domains in a way that guarantees deterministic transmission across non-TSN domains.

Table 1. Summary of related work.

Solution	Key Non-TSN Technology	Reference	Short Description
Wire-based	DetNet	[25]	Design an LFS mechanism that guarantees deterministic worst-case latency and zero packet loss for time-sensitive flows in Large-scale Deterministic Networks.
		[26]	Formulates the Deterministic Networking (DN) planning problem and presents an effective solution based on column generation and dynamic programming for Large-Scale Deterministic Networks.
	DIP	[11]	Proposes a hierarchical network containing access networks and a core network, in which access networks perform TAS to aggregate time-sensitive traffic and the core network exploits DIP to achieve long-distance deterministic transmission.
		[28]	Proposes the C-TSDN that enables cycle-based end-to-end deterministic transmission with multi-layer collaboration supported by the source routing and network slicing.
	SDN	[7]	Proposes an interconnection scheme following a software-defined TSN (SD-TSN) paradigm to enable deterministic communications among multiple closed networks.
	OTN	[29]	Proposes an SDN-enable optical transport network simulation platform to support the end-to-end service resource prejudge in cross-domain TSN.
Wireless-based	5G	[34]	Constructs a TSN-5G architecture, where the 5G system acts as a logical TSN-capable bridge, and then designs a Double Q-learning-based hierarchical particle swarm optimization algorithm (DQHPSO) to search for the optimal scheduling solution.
		[35]	Describes the recent standardization and developments in TSN and 5G and discusses the integration between TSN and 5G to enable bounded low-latency communications.
		[36]	Discusses current research and standardization work on 5G-TSN integration and quantifies, for a closed loop control application, the 5GS bridge delay.
	IEEE 802.11	[31]	Describes the approaches to extend TSN to wireless with IEEE 802.11, containing the wireless network management model, wireless time synchronization, time-aware scheduling, wireless link reliability, etc.
		[32]	Discusses the integration challenges of wired TSN and wireless local area network technologies and proposes a hybrid TSN device architecture.
		[33]	Presents a modular, multi-domain controller architecture to provide end-to-end TSN-enabled control over LAN and WLAN domains

3. Software-Defined Time-Sensitive Networking Framework

Following the paradigm of network control and data forwarding separation in SDN and the fully centralized configuration model of TSN [37], the SD-TSN framework consists of the physical infrastructure plane, the orchestration and configuration plane, and the application plane, as shown in Figure 1.

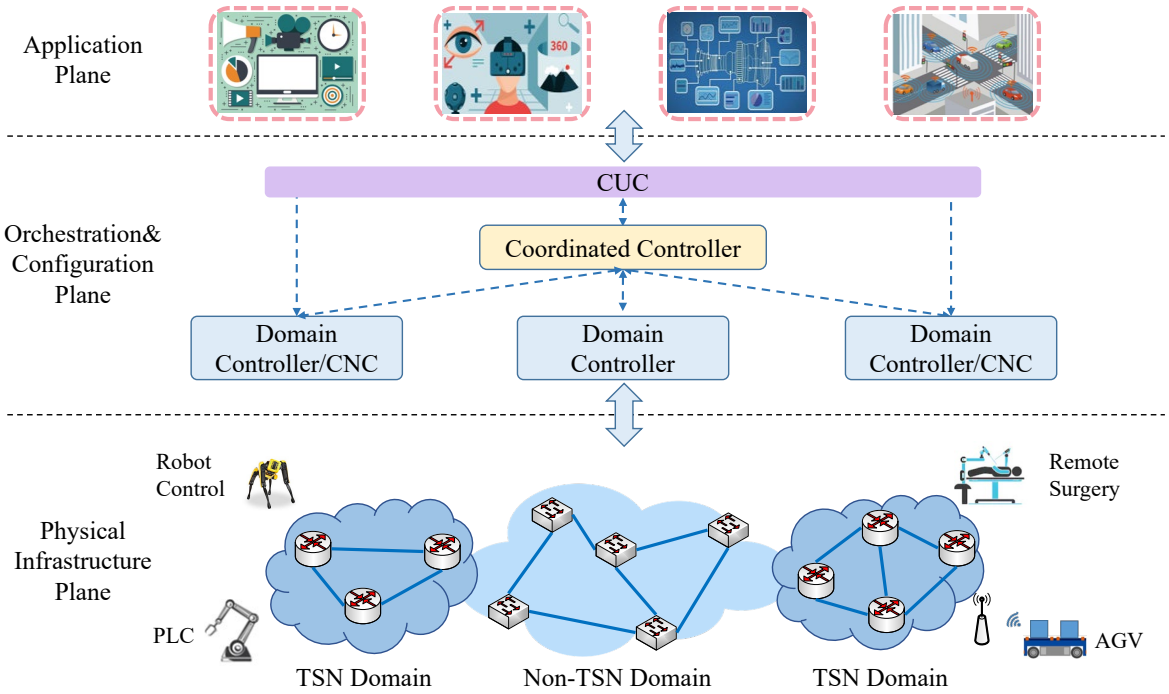


Figure 1. Illustration of the SD-TSN framework.

3.1. Physical Infrastructure Plane

The physical infrastructure plane includes TSN switches, non-TSN switches, and diverse end stations, such as the sensors, actuators, devices, and so on. TSN switches must support the forwarding process specified by IEEE 802.1Q (mainly including frame filtering, queuing frames, transmission selection, etc.) in a way that enables the implementation of the precise shaping and scheduling of time-sensitive traffic flows. These end stations will generate a variety of time-sensitive traffic flows, such as control data traffic from PLC to the robotic arms and cyclic data from controller to input/output communication. In SD-TSN, directly interconnected TSN-capable switches composed of a TSN domain and non-TSN domain are similar to this. We assume that the non-TSN domain is in charge of connecting different TSN domains, and each end station only connects with the TSN domain. There are several candidate technologies for the non-TSN domain, e.g., DetNet [38], Deterministic Dynamic Networks (DDNs) [39], DIP [40], 5G [34], and OpenFlow switches [7], etc.

3.2. Orchestration and Configuration Plane

This plane is logically centralized, consisting of the domain controllers, the coordinated controller, and the Centralized User Configuration (CUC). The domain controllers control the network domains, such as the Centralized Network Configuration (CNC) for the TSN domain. The coordinated controller communicates with all domain controllers, coordinating the network control and management orders among them. CUC communicates with end stations and the application plane, being responsible for retrieving the capabilities of end stations and transferring the requirements of the application plane to domain controllers or the coordinated controller.

The domain controller/CNC mainly consists of four modules, as described below:

- *Topology discovery:* This module exploits and collects the Link Layer Discovery Protocol (LLDP) to discover the network topology of the domain to support the path computation. Moreover, it sends LLDP messages periodically in a way that ensures the real-time discovery of the changes in topology, such as node addition and deletion.
- *Path computation:* Based on the topology information, it is responsible for computing the transmission path for each traffic flow according to the strategies from the coordi-

nated controller. Various routing algorithms can be used for path computation, such as the Dijkstra shortest-path algorithm.

- *Traffic scheduling:* It is in charge of generating schedules for the domain to indicate the transmission behavior of traffic flows, such that it guarantees determinism. Based on the requirements from the CUC and strategies from the coordinated controller, this module leverages scheduling algorithms to compute the schedules, such as integer linear programming (ILP)-based and heuristic-based algorithms.
- *Resource pool:* It records all flows' transmission behavior (e.g., sending time on the node) and the network state (e.g., bandwidth, throughput, and utilization) in the domain, and it updates them to the coordinated controller to form the global view.

The domain controllers communicate with the physical infrastructure through management protocols, such as NETCONF, SNMP, and RESTCONF [23].

The coordinated controller is composed of six main modules.

- *Resource management:* This module records the network resources from all domain controllers and formulates a global view of all network domains.
- *Topology management:* This module collects the topology information of each domain to build the topology of all network domains.
- *Path management:* It computes the path for the end-to-end connection across multiple domains based on the global topology information.
- *Connection management:* It builds the end-to-end connection for services across multiple network domains and manages the connections.
- *Time synchronization:* It manages and configures the time synchronization functions between multiple network domains.
- *Traffic policy:* Based on the global view of the network, it coordinates the transmission schedules among network domains.

The CUC translates and transfers the services requirements such as the deadline and other types of information to the domain controllers or coordinated controller for network configuration. When the CUC receives the service requirements, it identifies the end stations that generate and receive the traffic flows (namely, talkers and listeners) for each service. Then, the talkers group, listeners group, and requirements are sent to the domain and coordinated controllers to generate the transmission schedules for all traffic. After the schedules are computed, the CUC is in charge of informing the corresponding end stations according to the schedules. For instance, it classifies the services into different types with an identification (ID) and assigns sending time for each flow. Within the TSN domain, the service types correspond to the traffic classes, and the traffic classes are distinguished through the VLAN ID and priority code point (PCP).

3.3. Application Plane

The application plane covers an array of colorful and diverse applications, such as smart manufacturing and automotive networks. With the capability of the orchestration and configuration plane, operators can distribute the requirements through high-level language for various demands without caring about the underlying details. These service requirements are transferred to the CUC through standard Application Programming Interface (API), such as the RESTful API.

4. Multi-Domain Time-Aware Traffic Scheduling

In this section, we propose a multi-domain time-aware traffic scheduling model to generate transmissions schedule for each domain.

4.1. System Model

Consider an SD-TSN network in which two TSN domains are connected through a non-TSN domain, as shown in Figure 1. The network can be modeled as a graph $G^n = (V^n, E^n), n = 0, 1, 2$ where V^n is the set of network nodes in the nth network, and E^n

is the set of network links in corresponding networks. G^0 denotes the non-TSN domain and $G^n (n \geq 1)$ denotes the TSN domains. A link $[v_a^n, v_b^n]$ of E^n connects the source node v_a^n and the destination node v_b^n. We assume an equivalence between the port and its associated link since any egress port is connected to at most one link. Hence, a physical queue in source node v_a^n of the link $[v_a^n, v_b^n]$ is denoted as $q_m^{[v_a^n v_b^n]}$.

A set of periodic time-sensitive traffic flows F is considered. Each flow $f_i \in F$ is defined as

$$f_i = \{src_i, dst_i, dl_i, T_i, l_i\} \tag{1}$$

where src_i is the source node, dst_i is the destination node, dl_i is the deadline requirement, T_i is the period of the flow, and l_i is the flow length. The transmission path of f_i between its source and destination is denoted by $path_i$, containing an ordered sequence of nodes from src_i to dst_i. Within SD-TSN, it is split into several path segments, and each path segment is defined by sp_i^n. More specifically, $sp_i^{v_a^n}$ represents a portion of the path of f_i in G^n, which ends at v_a^n. For each path segment, the egress and ingress nodes are denoted as $v_{x_i}^n$ and $v_{y_i}^n$, respectively. In this paper, we assume the paths are known before the traffic scheduling.

In the transmission of a flow, the propagation delay and queuing delay on link $[v_a^n, v_b^n]$ of f_i are denoted as $pd^{[v_a^n, v_b^n]}$ and $qd_i^{[v_a^n, v_b^n]}$, respectively. The transmission delay and processing delay of flow f_i at node v_a^n are denoted as $td_i^{v_a^n}$ and $fd_i^{v_a^n}$, respectively. Denoting the bandwidth as B, we have $td_i^{v_a^n} = l_i/B$. Given that the propagation, processing, and transmission delays in the network elements are deterministic and constant [41], we have

$$cd_i^{[v_a^n, v_b^n]} = td_i^{v_a^n} + pd^{[v_a^n, v_b^n]} + fd_i^{v_b^n} \tag{2}$$

In TSN domains, the gate open time for flow f_i in queue $q_m^{[v_a^n, v_b^n]}$ is denoted as $ot_{i,m}^{[v_a^n, v_b^n]}$. The length of time that the gate is on the state of *open* is defined as the open window, which is denoted as $ow_{i,m}^{[v_a^n, v_b^n]}$. The cycle of the GCL, i.e, the interval over which the operations in a GCL repeats, is denoted as $gc^{[v_a^n, v_b^n]}$. It is computed as the least common multiple of periods of flows to be transmitted on link $[v_a^n, v_b^n]$. The notation details are shown in Table 2.

Table 2. Notations.

Symbol	Definition
G^n	the nth network domain
V^n	the set of network nodes in G^n
E^n	the set of network links in G^n
v_a^n	the ath network node in V^n
$[v_a^n, v_b^n]$	the link that connects source node v_a^n and destination node v_b^n
F	the set of time-sensitive traffic flow
f_i	the ith flow in F
dst_i	the destination node of f_i
dl_i	the deadline requirement of f_i
T_i	the period of f_i
l_i	the length of f_i
$path_i$	the path of f_i
sp_i^n	the path segment of f_i in G^n
$sp_i^{v_a^n}$	the path of f_i in G^n before v_a^n in G^n
$v_{x_i}^n$	the egress node of sp_i^n
$v_{y_i}^n$	the ingress node of sp_i^n
$F^{[v_a^n, v_b^n]}$	the set of time-sensitive traffic flows routed through link $[v_a^n, v_b^n]$
$pd^{[v_a^n, v_b^n]}$	the propagation delay on $[v_a^n, v_b^n]$
$td_i^{v_a^n}$	the transmission delay of f_i at v_a^n
$fd_i^{v_a^n}$	the processing delay of f_i at v_a^n
$qd_i^{[v_a^n, v_b^n]}$	the queuing delay of f_i on $[v_a^n, v_b^n]$

Table 2. *Cont.*

Symbol	Definition
$cd_i^{[v_a^n, v_b^n]}$	the sum of the transmission delay, processing delay, and propagation delay of f_i on $[v_a^n, v_b^n]$
ed_i^n	the end-to-end delay between the ingress node and the egress node of sp_i^n
B	the link bandwidth
$q_m^{[v_a^n, v_b^n]}$	the mth queue at v_a of $[v_a^n, v_b^n]$
$ot_{i,m}^{[v_a^n, v_b^n]}$	the open time of the gate for f_i
$ow_{i,m}^{[v_a^n, v_b^n]}$	the open window of the gate for f_i
$gc^{[v_a^n, v_b^n]}$	the gating cycle at node v_a^n of $[v_a^n, v_b^n]$
$t_{ig}^{[v_a^n, v_b^n]}$	the ingress timestamps on $[v_a^n, v_b^n]$
$t_{eg}^{[v_a^n, v_b^n]}$	the egress timestamps on $[v_a^n, v_b^n]$
δ_i	the difference in transmission delay between f_i and f_0.
θ_i^0	the worst-case end-to-end delay of f_i on sp_i^0
st_i^n	the starting time of f_i in G^n

4.2. Problem Formulation

In the end-to-end transmission of a time-sensitive traffic flow, the end-to-end delay can be divided into four parts, namely, the propagation delay, the transmission delay, the processing delay, and the queuing delay. The first three delays can be calculated directly based on the global view and, thus, can be considered to be deterministic. Queuing occurs in a switch when multiple traffic flows attempt to transmit through the same output port simultaneously [42]. In real scenarios, a wide variety of traffic flows will arrive at the switch at any time and demand to be transmitted; thus, the queuing delay is non-deterministic. TSN enhances the hardware mechanism of Ethernet-based switches and employs various shaping and scheduling techniques to achieve bounded queuing delays in TSN domains. However, time-sensitive traffic flows may suffer non-deterministic queuing delays when they are transmitted in non-TSN domains.

SD-TSN aims to bound the non-deterministic queuing delay to provide deterministic guarantees. It can be achieved as long as no traffic flows transmit over the same output port at the same time in non-TSN domains, i.e., the queuing delay of these flows on each node in the non-TSN domain is equal to zero. To this end, SD-TSN exploits traffic shaping and scheduling to allocate the time for all traffic to be injected into the non-TSN domains in the TSN domain before the time-sensitive traffic enters the non-TSN domain. With the allocated injection times, all flows do not interfere with each other during the transmission in non-TSN domains. However, traffic outside the schedule (e.g., best-effort traffic) will inevitably appear in the non-TSN domain which impairs the deterministic guarantees. For this issue, SD-TSN exploits the logical centralized control to restrict and compensate for the non-deterministic queuing delay caused by these flows. Specifically, we define the following constraints to guarantee deterministic communication in the SD-TSN.

In the TSN domain, the scheduling model leverages GCLs to specify flow transmission behavior for determinism. For each gate event of all GCLs, it must be able to accommodate the transmission of corresponding flows, which can be describe as follows:

$$\begin{aligned} &\forall G^n, n \geq 1, \forall [v_a^n, v_b^n] \in E^n, \forall f_i \in F^{[v_a^n, v_b^n]} : \\ &\frac{l_i}{B} \leq ow_{i,m}^{[v_a^n, v_b^n]} \leq gc^{[v_a^n, v_b^n]}. \end{aligned} \tag{3}$$

In addition, the interval between gate open events on two adjacent nodes must allow for the normal transmission of time-sensitive flows. A prematurely closed gate will accidentally terminate the flow transmission with adverse effect. Thus, we have

$$\begin{aligned} &\forall G^n, n \geq 1, \forall [v_a^n, v_b^n] \in E^n, \forall f_i \in F^{[v_a^n, v_b^n]} : \\ &ot_{i,m}^{[v_b^n, v_b^n+1]} \geq ot_{i,m}^{[v_a^n, v_b^n]} + cd_i^{[v_a^n, v_b^n]}. \end{aligned} \tag{4}$$

If gates of more than one physical queue are *open*, the interference of the flows can occur. For example, when a flow in the queue with a lower priority is transmitting, flows in other queues wait until a frame in that flow finishes transmitting. In addition, the flow transmission of the queue with a lower priority is interrupted. To avoid this phenomenon among multiple physical queues, the open windows of their gates are not allowed to overlap. That is,

$$\begin{aligned} &\forall G^n, n \geq 1, \forall [v_a^n, v_b^n] \in E^n, \forall f_i, f_j \in F^{[v_a^n, v_b^n]}, i \neq j, \\ &\forall \alpha \in \left[0, gc^{[v_a^n, v_b^n]} / T_i - 1\right], \forall \beta \in \left[0, gc^{[v_a^n, v_b^n]} / T_j - 1\right] : \\ &(ot_{i,m}^{[v_a^n, v_b^n]} + \alpha \times T_i \geq ot_{j,m'}^{[v_a^n, v_b^n]} + ow_{j,m'}^{[v_a^n, v_b^n]} + \beta \times T_i) \vee \\ &(ot_{j,m'}^{[v_a^n, v_b^n]} + \beta \times T_j \geq ot_{i,m}^{[v_a^n, v_b^n]} + ow_{i,m}^{[v_a^n, v_b^n]} + \alpha \times T_i) \end{aligned} \tag{5}$$

For each time-sensitive traffic flow, it is required to arrive at the destinations within the deadline, then we have

$$\forall f_i \in F : 0 < ot_{i,m}^{[dst_i - 1, dst_i]} + cd_i^{[dst_i - 1, dst_i]} - ot_{i,m}^{[src_i, src_i + 1]} + cd_i^{[src_i, src_i + 1]} \leq dl_i \tag{6}$$

For network domains that do not support the hardware TSN mechanism, we aim to ensure that all time-sensitive traffic flows do not interfere with each other during the transmission in these domains by allocating isolated time slots for them when they enter the domain. First of all, the injection time of each time-sensitive flow into the non-TSN domain should not overlap with each other to prevent adverse results. With TAS, the constraint can be described as follows:

$$\begin{aligned} &\forall G^n, n \geq 1, \forall f_i, f_j \in F, v_{x_i}^n = v_{x_j}^n, i \neq j, \\ &\forall \alpha \in \left[0, gc^{[v_{x_i}^n, v_{y_i}^0]} / T_i - 1\right], \forall \beta \in \left[0, gc^{[v_{x_j}^n, v_{y_j}^0]} / T_j - 1\right] : \\ &(ot_{i,m}^{[v_{x_i}^n, v_{y_i}^0]} + \alpha \times T_i \geq ot_{j,m'}^{[v_{x_j}^n, v_{y_j}^0]} + ow_{j,m'}^{[v_{x_j}^n, v_{y_j}^0]} + \beta \times T_i) \vee \\ &(ot_{j,m'}^{[v_{x_j}^n, v_{y_j}^0]} + \beta \times T_j \geq ot_{i,m}^{[v_{x_i}^n, v_{y_i}^0]} + ow_{i,m}^{[v_{x_i}^n, v_{y_i}^0]} + \alpha \times T_i) \end{aligned} \tag{7}$$

Incorporating the global view of the coordinated controller, the propagation delay, transmission delay, processing delay, and transmission path are known. Thus, the goal of transmission with zero queuing delay can be achieved by further adjusting the departure time of the flow at the egress node in the TSN domain; thus, the constraint is as follows:

$$\begin{aligned} &\forall \left[v_a^0, v_b^0\right] \in E^0, \forall f_i, f_j \in F^{[v_a^0, v_b^0]}, i \neq j \\ &\forall \alpha \in [0, hp(i,j)/T_i - 1], \forall \beta \in \left[0, hp(i,j)/T_j - 1\right] : \\ &(st_i^0 + \sum_{k \in sp_i^{v_a^0}} cd_i^{[k,k+1]} + td_i^{v_a^0} + \alpha \times T_i \geq st_j^0 + \sum_{k \in sp_j^{v_a^0}} cd_j^{[k,k+1]} + \beta \times T_j) \vee \\ &(st_j^0 + \sum_{k \in sp_j^{v_a^0}} cd_j^{[k,k+1]} + td_j^{v_a^0} + \beta \times T_j \geq st_i^0 + \sum_{k \in sp_i^{v_a^0}} cd_i^{[k,k+1]} + \alpha \times T_i) \end{aligned} \tag{8}$$

where $st_i^0 = ot_{i,m}^{[v_{x_i}^n, v_{y_i}^0]} + cd_i^{[v_{x_i}^n, v_{y_i}^0]}$, representing the starting time of flow f_i in a non-TSN domain, and $hp(i,j) = lcm(T_i, T_j)$ is the least common multiple of f_i and f_j, also known as the hyperperiod of f_i and f_j.

Considering the influence of unscheduled best-effort traffic flows, we use the meter mechanism to limit the maximum forwarding traffic at the ingress port. In the worst case, a time-sensitive traffic flow waits in a non-TSN node until the best-effort traffic is transmitted. Denote the maximum traffic volume allowed for forwarding in one gate control period as ρ, the volume of best-effort traffic can vary between 0 and ρ at each non-TSN node. Then, the queuing can occur randomly, resulting in a random end-to-end delay in the non-TSN domain. In the proposed model, we utilize the mechanism in [43] to measure the delay in the non-TSN domain. In the non-TSN domain, the flow for the delay measurement, denoted as f_0, is sent periodically. The period of f_0 is named as a test cycle. The ingress timestamps $t_{ig}^{[v_a^n-1, v_a^n]}$ and the egress timestamps $t_{eg}^{[v_a^n, v_a^n+1]}$ for each network node v_a^n are recorded and constructed as the timestamp pairs $(t_{ig}^{[v_a^n-1, v_a^n]}, t_{eg}^{[v_a^n, v_a^n+1]})$. The timestamp pairs are collected by the domain controller and sent to the coordinated controller to construct a delay view. With timestamp, the delay of path segment sp_i^0 in the hth test cycle is given as

$$ed_{i,h}^0 = t_{eg,h}^{[v_{x_i}^0, v_{x_i}^0+1]} - t_{ig,h}^{[v_{y_i}^0-1, v_{y_i}^0]} - \delta_i, \tag{9}$$

where $\delta_i = \sum_{k \in sp_i^0} (cd_i^{[k,k+1]} - cd_0^{[k,k+1]})$ reflects the difference in constant delay between f_i and f_0. Furthermore, the worst-case delay of flow f_i in the non-TSN domain is given as

$$\theta_i^0 = max\{ed_{i,h}^0 | h = 1, 2, \cdots, H\}. \tag{10}$$

When a time-sensitive traffic flow is transmitted from the non-TSN domain to a TSN domain, the random queuing delay in the non-TSN domain results in a delay variation between the flow periods. To eliminate the delay variation, the departure time of flow f_i at the ingress switch of the TSN domain has the following constraint:

$$\begin{aligned} &\forall f_i \in F, \forall [v_{x_i}^n, v_{y_i}^0], [v_{x_i}^0, v_{y_i}^{n'}], n \geq 1, n' \geq 1 : \\ &ot_{i,m}^{[v_{y_i}^{n'}, v_{y_i}^{n'}+1]} \geq st_i^0 + \theta_i^0 + pd^{[v_{x_i}^0, v_{y_i}^{n'}]} + fd_i^{v_{y_i}^{n'}}. \end{aligned} \tag{11}$$

4.3. Scheduling Algorithm

This subsection elaborates on the implementation of the proposed scheduling model. The most straightforward method to compute the schedule is simply building constraints all at once and then putting them into an SMT solver. However, this approach is highly time-consuming and even fails especially in the case of the number of flows being large. Hence, we present a Tabu search-based multiple-domain scheduling (TMDS) Algorithm 1 for the proposed scheduling model. Compared to the SMT-based method, our method aims to find a feasible solution for flows one by one, rather than solving all constraints together for all flows, and the pseudocode of the algorithm is shown in Algorithm 1. Here, we simply create an equivalence between the sending time and the GCL since the transformation is straightforward [44].

Algorithm 1 Tabu search-based multiple-domain scheduling algorithm

Input: Flow set F, Network domain set G^n
Output: Schedule set S

1: $S \leftarrow \{\}$
2: $F^* \leftarrow SortFlow(F)$ ▷ Sort F by flow period in ascending order
3: **for all** $f_i \in F^*$ **do**
4: Get worst-case delay θ_i^0 according to Equation (10)
5: $S_i^* \leftarrow InitialSchedule(f_i, S, \theta_i^0)$ ▷ Initial the earliest sending time of f_i on nodes over the path
6: $TC \leftarrow TestConflict(G^n, f_i, S, S_i^*)$
7: **while** $TC \neq \varnothing$ **do**
8: $NeighborS_i^* \leftarrow GenerateNeighborSolution(S_i^*, TC)$
9: $NeighSolution \leftarrow \{\}$
10: **for all** $NS_i^* \in NeighborS_i^*$ **do**
11: $NeighSolution \leftarrow TestConflict(G^n, f_i, S, NS_i^*) + NeighSolution$
12: **end for**
13: $S_i^* \leftarrow BestSolution(NeighSolution)$
14: $TC \leftarrow TestConflict(G^n, f_i, S, S_i^*)$
15: **end while**
16: $S \leftarrow S + S_i^*$
17: **end for**
18: **return** S

As shown in Algorithm 1, all flows are first sorted by period from frequent to infrequent (line 2), and the larger flow will be in the former when the periods of flows are equal. This is because larger and more frequent flows are harder to be scheduled later. Then, it initializes S_i^* for f_i through $InitialSchedule()$ (line 5), that is, a set of sending times of f_i on all nodes over the path. A set of sending times can be computed by the starting time at the source node according to the proposed constraints, and $InitialSchedule()$ assigns an earliest possible starting time for f_i according to the existing S. Next, $TestConflict()$ tests the validity of S_i^* by putting it into G^n (line 6). Each conflict generated during testing can be regarded as a duration, e.g., if the f_i with S_i^* and the scheduled flows with S have a conflict according to Equation (5), the conflict value is equal to the length of the overlap between f_i and the scheduled flows. If TC satisfies all constraints, S_i^* will be merged into S to update the schedule (line 16). Otherwise, the algorithm generates the neighborhood of the schedule according to the conflict, i.e., offset the initial starting time by the conflict (line 8). Each solution in the neighborhood is verified to comply with the constraints through $TestConflict()$, and the best solution is selected for the next iteration (lines 10–14). This procedure will iterate until a feasible S_i^* is found. The abovementioned steps are repeated until all flows have been scheduled.

5. Performance Evaluation

5.1. Evaluation in Simulated Environments

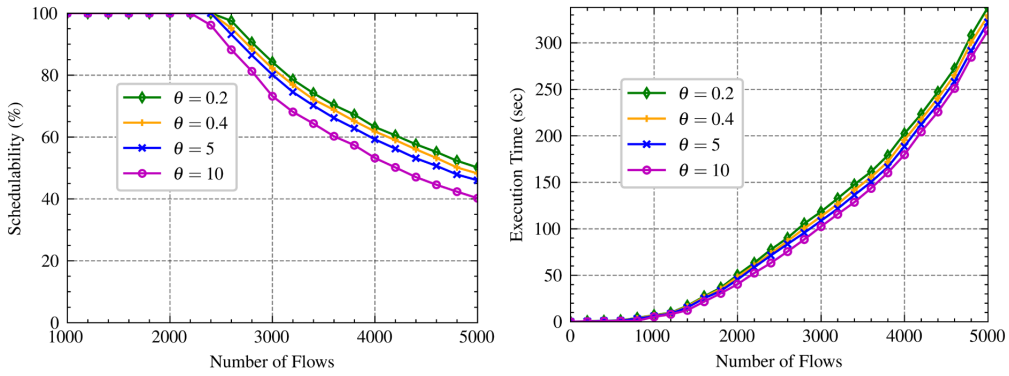
5.1.1. Simulation Setup

In this section, we evaluate the performance of our approach and compare it with the widely-applied SMT approach [16]. The evaluations are mainly focused on the schedulability and execution time, where schedulability is defined as the ratio between the number of scheduled flows and the number of flows to be scheduled. The simulation topology is the same as that in Figure 1. The bandwidth of the network is 1 Gbps. Each time-sensitive traffic flow transmits from a TSN domain to another TSN domain, and the non-TSN domain is composed of non-TSN capable switches. The propagation delay is set as 0.5 µs, and the forwarding delay is set as 1 µs. The period of the time-sensitive traffic is selected randomly from the set {1 ms, 2 ms, 4 ms, 8 ms, 16 ms, 32 ms, 64 ms}. Each time-sensitive traffic flow is required to finish its transmission in one period. The frame length is 64 bytes, and a flow contains γ frames, γ = 1, 2, 4, or 8 as in [20]. The time limit for scheduling approaches is

20 min [45]. For each flow, the value of γ is selected randomly with uniform distribution. These scheduling approaches were implemented in Python 3.10 on a PC with 64-bit 8-core 3.0GHz Intel Core-i7 and 16 GB memory.

5.1.2. Simulation Results

Various WCDs in the non-TSN domain: Figure 2a compares the schedulability of the TMDS under different WCDs. The WCD in the non-TSN domain θ is set as 0.2 ms, 0.4 ms, 5 ms, and 10 ms. The results show the effectiveness of the proposed model in different WCDs, and the smaller θ has an advantage in the schedulability. If the flow period is similar to the worst-case delay in the non-TSN domain, the flow is less possible to be scheduled. Moreover, the smaller θ can bring the larger delay bounds in the TSN domains. A larger delay bound means the larger queuing delay is tolerant, which means more flows can be scheduled when they transmit through the same link. Figure 2b presents the relationship of the execution time and the flow number. It shows that the more the number of flows increases, the more the execution time increases. This is because the excessive flow will lead to a much larger neighborhood space, which will lead to a much larger computation time. Moreover, the execution time is the longest when $\theta = 0.2$, since the schedulable flow is the highest. Considering that different WCDs generate results with similar trends, we assume θ = 10 ms for the following tests.



(a) (b)

Figure 2. Simulation results with different WCDs. (**a**) Schedulability versus flow number; (**b**) Execution versus flow number.

Various flow numbers: A comparison of the schedulability for two scheduling approaches is shown in Figure 3a. The SMT-based approach can schedule all flows when the number of flows is less than 2200 and then decrease as the flow number increases. This is because the runtime by it reaches the time limit when scheduling about 2151 flows, i.e., no more new flows can be scheduled as the flow number increases. Our approach can maintain 100% schedulability until 2200 flows and then decrease similarly thereafter. Note that the downward trend of TMDS is slightly faster than that of the SMT-based after the flow number larger than 2200. This is due to the fact that TMDS attempts to find a feasible solution for flows one by one, which may yield suboptimal solution in terms of the whole flows. Figure 3b compares the execution time of these approaches for scheduling various flow numbers. Because all flows in a hyper-period of flow periods need to be computed according to given constraints, the SMT-based approach takes a longer time to schedule all flows. For instance, it spends approximately 1042 s to schedule 2000 flows; meanwhile, the runtime of our approach is only 3.84% of it, i.e., about 40 s. For the same reason, the SMT-based technique hits the time limit when scheduling approximately 2151 flows. The TMDS tries to search for a feasible solution through a Tabu search method-based trial rather than calculating all flows in a hyper-period according to the given constraints, which significantly reduces the execution time, e.g., it spends about 313 s when the flow number is 5000. The comparison of the bandwidth utilization for two scheduling approaches is given in Figure 3c. As expected, the bandwidth utilization

increases as the flow number increases when the flow number is less than 2200. After that, the bandwidth utilization of TMDS gradually decreases as the flow increases since the schedulable flow decreases gradually. It is worth noting that the maximum bandwidth utilization is less than 10% by two approaches, which reserves sufficient resources for the transmission of BE traffic. Furthermore, the time-sensitive traffic flows are typically periodic and small according to the descriptions of flow properties in IEC/IEEE 60802 [46]; thus, their transmission typically would not block BE traffic by occupying the egress port for a long time.

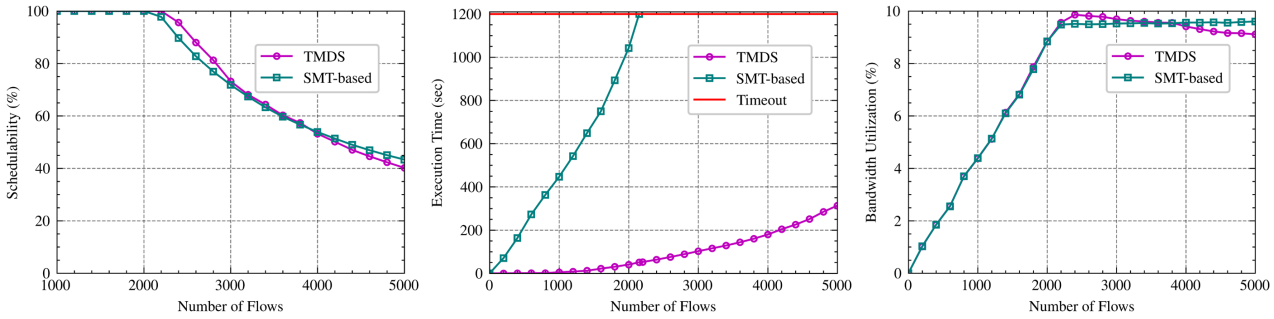


Figure 3. Simulation results when θ = 10 ms. (**a**) Schedulability; (**b**) Execution time; and (**c**) Bandwidth utilization versus flow number, respectively.

Various bandwidth utilizations: To further evaluate the robustness of our approach, we design a scheduling test in various bandwidth utilizations. Specifically, we add a certain number of time-sensitive flows to the networks with different bandwidths, respectively. We choose 10 different bandwidth utilizations of the networks: 0%, 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%, 100%. The number of added flows is randomly generated between 1800 and 2000, and these flows need to be scheduled by two scheduling approaches. Figure 4a compares the schedulability of two approaches under different network bandwidths. It shows that both scheduling approaches can schedule all flows when the remaining resources are sufficient. For instance, the TMDS and the SMT-based approach can maintain 100% schedulability before the bandwidth utilization less than 60% and 70%, respectively. Furthermore, the schedulability by TMDS is slightly lower than the SDM-based after the bandwidth utilization greater 60%. This is because the TMDS following a heuristic strategy may miss some feasible solutions, making it more sensitive to the remaining resources. The comparison of execution time for scheduling approaches is shown in Figure 4b. The TMDS only takes an average of 32 s to schedule these flows while the SMT-based approach spends around 907 s for scheduling. Overall, the TMDS drastically reduces execution time while ensuring that schedulability is not degraded.

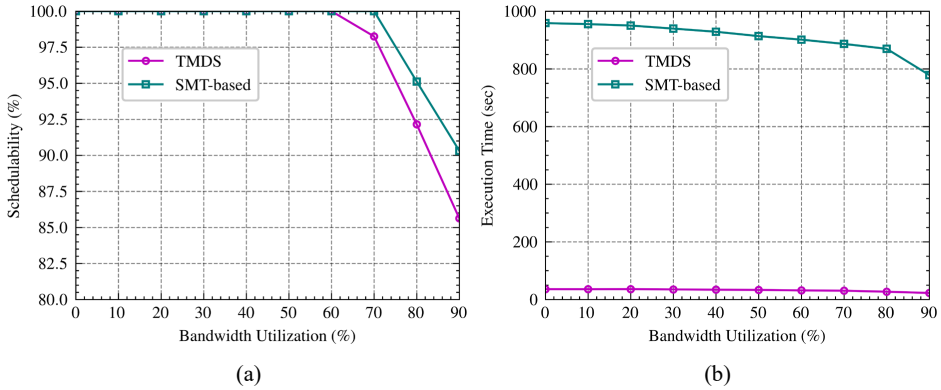


Figure 4. Simulation results when θ = 10 ms. (**a**) Schedulability and (**b**) Execution time under different bandwidth utilizations.

5.2. Evaluation on Testbed in Real Environment

5.2.1. Experimental Setup

In this subsection, we present the prototype testbed we built to verify the effectiveness of the proposed approach in a realistic environment, as shown in Figure 5. The testbed is composed of two parts, which are the control plane for centralized control by the controller and the data plane for transmitting the time-sensitive flows, respectively. In the control plane, we develop the CNC for the TSN domain on a server with an Intel Xeon E2224 processor (Intel, Beijing, China) running at 3.4 GHz and 16 GB RAM. The proposed model is inserted into the traffic scheduling module for the GCLs' computation. The OpenDaylight is deployed as the non-TSN domain controller to configure flow tables and meter mechanism in OpenFlow switches. In the data plane, we assume two TSN domains (Domain A and Domain C) connected by a non-TSN domain (Domain B that is composed of OpenFlow switches) to enable transmission across a non-TSN domain. All switches in the TSN domain support IEEE 802.1Qbv and IEEE 802.1AS for precise traffic scheduling and time synchronization. Two TSN domains are synchronized through the cross-domain time synchronization scheme while the TSN switches in each domain are synchronized through IEEE 802.1AS [47]. Table 3 shows the details of the hardware devices.

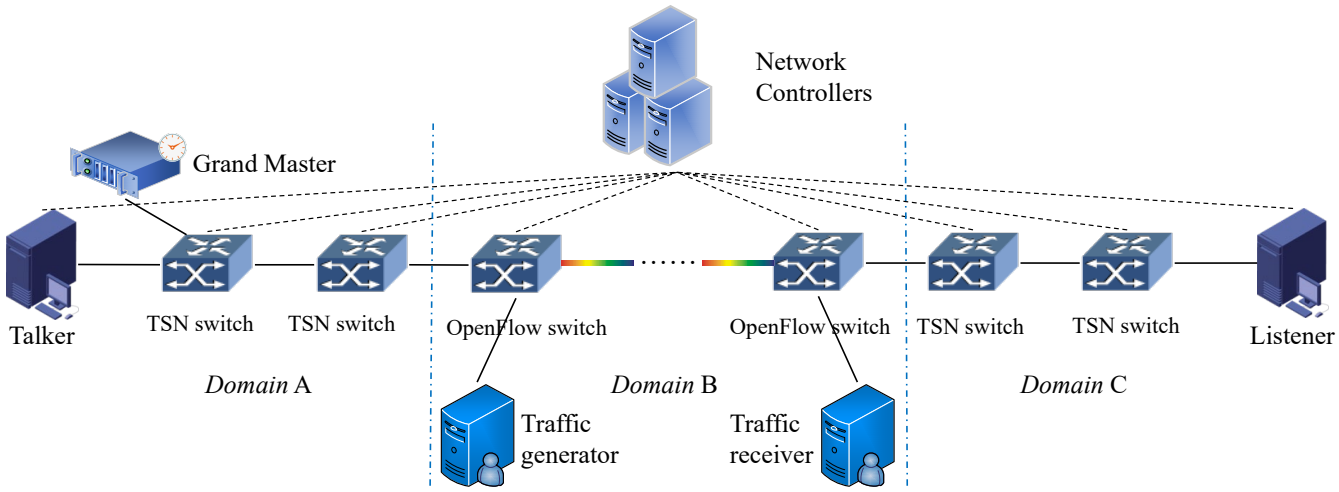


Figure 5. Illustration of the SD-TSN testbed.

Table 3. Hardware devices and configuration.

Unit	Module	Configuration
TSN switch	ETX TSN Switch (Beijing University of Posts and Communications, Beijing, China)	NETCONF, GE
OpenFLow switch	Centec V350 (Centec, Suzhou, China)	Openflow1.3, GE
Grand master	OSA 5421 (Oscilloquartz, Neuchâtel, Switzerland)	GE, Rubidium
Talker/Listner	VIAVI MTS 5800 (VIAVI, Beijing, China)	GE

In the experiment, we assumed all time-sensitive flows were sent from the talker to the listener, and the flow properties were the same as those in the previous subsection. Considering that typical industrial networks are usually based on three topologies: linear, ring, and tree [48]. We selected the linear topology as the test topology to evaluate our approach to avoiding the optimization space brought by various routing methods, and other topologies can be denoted as a set of linear topologies. In domains A and C, two TSN switches are linearly linked via a 1 Gbps full-duplex Ethernet cable. The traffic generator sends background traffic to the traffic receiver. The background traffic includes two parts, constant-rate traffic of 800 Mb/s and burst traffic of 200 Mb/s. All parts are assigned with the lowest priority, and the burst occurs randomly in the experiment. The propagation

delay, the processing delay, and the worst-case delay in the non-TSN domain are measured through the measurement mechanism in [43].

Two test cases were designed to measure the end-to-end delay and delay variation of the time-sensitive traffic flows. The delay variation is defined as the end-to-end delay offset relative to the minimum delay in all periods. All test traffic flows were transmitted from the talker to the listener. In Case 1, two OpenFlow switches are connected linearly with fiber in domain B. In Case 2, twenty-six OpenFlow switches are linearly connected through fiber in domain B, and the length of each fiber segment is 80 km. Thus, the distance of the non-TSN domain in case 2 is equal to 2000 km.

5.2.2. Experimental Results

Table 4 illustrates the experimental results of our approach with the increasing flow. It shows that the controller with the TMDS can respond to changes in traffic within one second, which is applicable to most industrial scenarios [9]. The response time in Case 2 is slightly longer than that in Case 1. This is because the messages take more time in domain B, thus increasing the time for the measurement of the WCD. The bandwidth utilization of these approaches increases as the flow number increase, but all are less than 5.03%. Therefore, the transmission of these flows would not interfere with or block the transmission of background traffic. In addition, the controller can schedule all flows for each test in Cases 1 and 2. For simplicity, we selected 50 flows for the next experiments.

Table 4. Experimental results on the testbed.

Approach	Flow Number	Case 1			Case 2		
		Schedulability (%)	Response Time (ms)	Bandwidth Utilization (%)	Schedulability (%)	Response Time (ms)	Bandwidth Utilization (%)
TMDS	10	100	79.602	0.47	100	181.201	0.51
	30	100	139.033	1.37	100	242.589	1.36
	50	100	271.356	2.35	100	379.265	2.41
	70	100	334.203	3.38	100	445.621	3.4
	90	100	413.025	5.01	100	528.624	4.98
SMT-based	10	100	1025.621	0.51	100	1159.325	0.5
	30	100	4253.625	1.38	100	4401.389	1.38
	50	100	7358.251	2.33	100	7468.256	2.39
	70	100	10,254.163	3.4	100	10,373.673	3.43
	90	100	15,425.025	5.03	100	15,626.674	5.02

Figure 6 shows the end-to-end delay and delay variation of the flow in Case 1. The worst-case delay in the SDN domain is about 780 µs according to the delay measurement mechanism. In this condition, the burst best-effort traffic flows result in congestion. The minimum queuing delay result from the best-effort traffic flows in the SDN domain is about 12.2 µs when the time-sensitive traffic flow waits for a frame to be transmitted. The maximum end-to-end delay in Case 1 is below 0.867 ms, demonstrating that the computed schedules achieved the bounded end-to-end delay. The delay variation in Case 1 is less than 1 µs, which is much more smaller than the delay variation results from the best-effort traffic flows. The maximum delay variation is about 1/1000 of the end-to-end delay, showing the low delay variation feature of the proposed method.

Figure 7 shows the end-to-end delay and delay variation of Case 2. The results show that the proposed model is effective in the long-distance test case. The maximum end-to-end delay is less than 10.97 ms, which is bounded, although much larger than that in Case 1 due to the propagation delay in the SDN domain. The delay variation in Case 2 is less than 10 µs, and the maximum delay variation is also about 1/1000 of the end-to-end delay. Although the delay variation is ten times that in Case 1, it is still much smaller than the delay variation results from the best-effort traffic flows in the SDN

domain. This demonstrates the capability of the proposed method in alleviating delay variation. The delay variations in Case 2 mostly derive from time errors between two TSN networks. According to the time-synchronization measurement, the maximum time error of the end-to-end time synchronization in Case 1 is 300 ns, while it is 8 µs in Case 2.

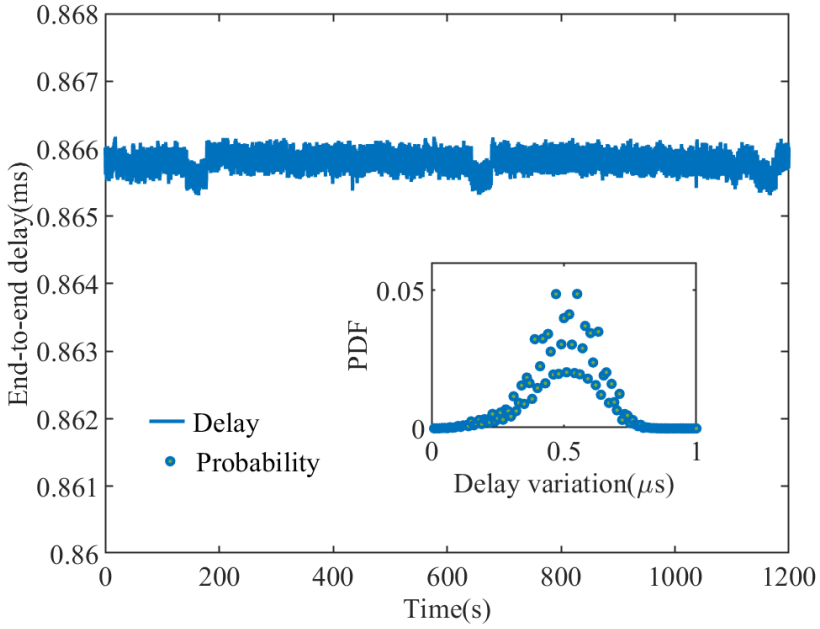


Figure 6. End-to-end delay and PDF of delay variation in Case 1, in which two OpenFlow switches are directly connected through fiber in domain B.

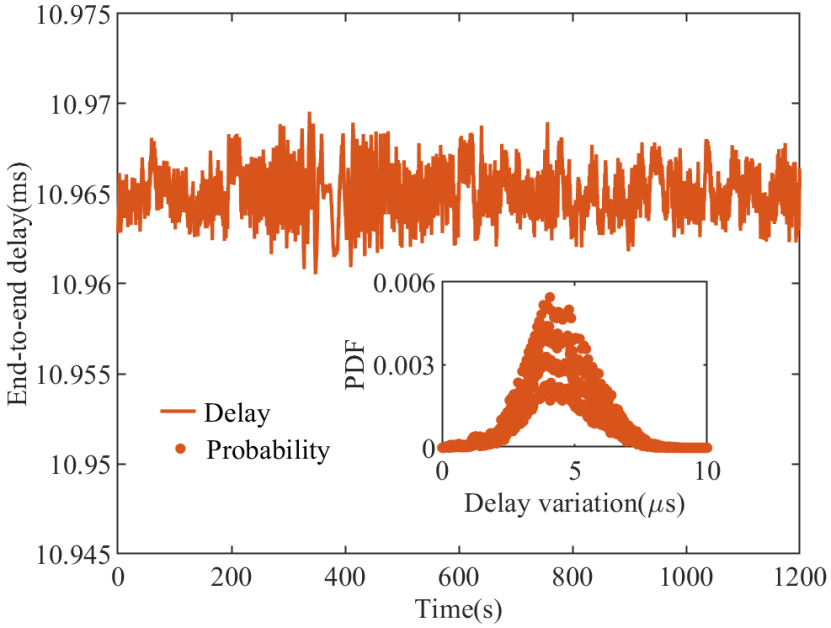


Figure 7. End-to-end delay and PDF delay variation in Case 2, in which twenty-six OpenFlow switches (Centec, Suzhou, China) are linearly connected over fiber in domain B. The distance between every two switches is 80 km; thus, the distance through the non-TSN domain is equal to 80 × 25 = 2000 km.

6. Conclusions and Future Works

The emerging industrial applications makes it imperative to guarantee the determinism of the end-to-end connection among multiple domains. In this paper, we propose a multi-domain time-aware traffic scheduling model in an SD-TSN framework. SD-TSN is designed based on the SDN paradigm in which the coordinated controller performs the cooperation among network domains. The proposed scheduling model unifies schedule time-sensitive traffic flows by harmonizing TAS in TSN domains and time-slots in non-TSN domains. The experimental results demonstrate that our proposed method can guarantee bounded delay and low delay variations for end-to-end transmission across non-TSN domains. The delay variation in a 2000 km end-to-end connection is below 10 µs.

However, the deployment of SD-TSN in the real environment still faces several challenges. First, the realistic application scenarios are typically flexible, dynamic, and involve many devices from different vectors, making WCD measurement a difficult task. A direct approach to this issue is to enhance the capability of the controller to collect timestamp information from individual nodes with a high frequency. This provides real-time WCD measurements to cope with complex and dynamic applications. However, too frequently, measurement message interactions will interfere with the transmission of other time-sensitive traffic, such as network control data and safe-critical data. Moreover, enhancing diverse low-level devices (e.g., sensors and actuators) enables them to support frequent interactions, which is not feasible in terms of feasibility or economically. Thus, a promising solution is a more refined design of domain controllers for non-TSN domains. For non-TSN domains with delay guarantees (e.g., 5G URLLC), the domain controller can straightforwardly measure the WCD. For other non-TSN domains, it adapts advanced techniques to measure the WCD without high-frequency interactions, e.g., relaxing the accuracy of WCD measurements and predicting WCD through artificial intelligence techniques.

On the other hand, time synchronization is a factor that cannot be ignored since it directly affects the performance of traffic scheduling. For similar reasons, it is a trade-off problem between performance and costs. SD-TSN needs to find a feasible solution for better deployment, e.g., maintaining a low synchronization precision while enabling traffic scheduling. Furthermore, the downgrade deployment of SD-TSN is also an open issue. Our work is dedicated to guarantee deterministic transmission across non-TSN domains. However, in real environments, the talkers or listeners likely directly connect to the non-TSN domains, and their requirements are not demanding at the same time. The proposed scheduling model is not suitable for this case, but SD-TSN has a high potential to cope with this issue based on enhancements to the capabilities of the control plane. The key challenge lies in the design of the scheduling model and corresponding algorithm, which is the focus of future work.

Author Contributions: Conceptualization, M.G. and G.S.; methodology, M.G. and G.S.; software, M.G.; validation, M.G.; formal analysis, M.G.; investigation, M.G.; resources, M.G. and G.S.; data curation, M.G.; writing—original draft preparation, M.G.; writing—review and editing, M.G. and G.S.; visualization, M.G.; supervision, G.S.; project administration, G.S., Y.L. and Y.H.; funding acquisition, G.S., Y.L. and Y.H. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by National Key R&D Program of China (Grant No. 2022YFC3803700).

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

TSN	Time-Sensitive Networking
SDN	Software-Defined Networking
SD-TSN	Software-Defined Time-Sensitive Networking
TAS	Time-Aware Shaper
BE	Best-Effort
GCL	Gate Control List
IT	Information Technology
OT	Operational Technology
DetNet	Deterministic Networking
LDN	Large-scale Deterministic Network
DIP	Deterministic IP
URLLC	Ultra-Reliable and Ultra-Low Latency Communications
SRv6	Segment Routing IPv6
DS-TT	Device-Side TSN Translator
NW-TT	Network-Side TSN Translator
DDN	Deterministic Dynamic Networks
CUC	Centralized User Configuration
CNC	Centralized Network Configuration
LLDP	Link Layer Discovery Protocol
ILP	Integer Linear Programming
SMT	Satisfiability Modulo Theories
WCD	Worst-Case Delay
PCP	Priority Code Point
VLAN	Virtual Local Area Network
API	Application Programming Interface

References

1. Wollschlaeger, M.; Sauter, T.; Jasperneite, J. The future of industrial communication: Automation networks in the era of the internet of things and industry 4.0. *IEEE Ind. Electron. Mag.* **2017**, *11*, 17–27. [CrossRef]
2. Yang, H.; Alphones, A.; Zhong, W.D.; Chen, C.; Xie, X. Learning-based energy-efficient resource management by heterogeneous RF/VLC for ultra-reliable low-latency industrial IoT networks. *IEEE Trans. Ind. Inform.* **2019**, *16*, 5565–5576. [CrossRef]
3. Moyne, J.R.; Tilbury, D.M. The emergence of industrial control networks for manufacturing control, diagnostics, and safety data. *Proc. IEEE* **2007**, *95*, 29–47. [CrossRef]
4. Atallah, A.A.; Hamad, G.B.; Mohamed, O.A. Routing and scheduling of time-triggered traffic in time-sensitive networks. *IEEE Trans. Ind. Inform.* **2019**, *16*, 4525–4534. [CrossRef]
5. Bruckner, D.; Stănică, M.P.; Blair, R.; Schriegel, S.; Kehrer, S.; Seewald, M.; Sauter, T. An introduction to OPC UA TSN for industrial communication systems. *Proc. IEEE* **2019**, *107*, 1121–1131. [CrossRef]
6. Bello, L.L.; Steiner, W. A perspective on IEEE time-sensitive networking for industrial communication and automation systems. *Proc. IEEE* **2019**, *107*, 1094–1120. [CrossRef]
7. Xue, J.; Shou, G.; Li, H.; Liu, Y. Enabling deterministic communications for end-to-end connectivity with software-defined time-sensitive networking. *IEEE Netw.* **2022**, *36*, 34–40. [CrossRef]
8. Leonardi, L.; Bello, L.L.; Patti, G. Bandwidth partitioning for Time-Sensitive Networking flows in automotive communications. *IEEE Commun. Lett.* **2021**, *25*, 3258–3261. [CrossRef]
9. Nasrallah, A.; Thyagaturu, A.S.; Alharbi, Z.; Wang, C.; Shao, X.; Reisslein, M.; ElBakoury, H. Ultra-low latency (ULL) networks: The IEEE TSN and IETF DetNet standards and related 5G ULL research. *IEEE Commun. Surv. Tutor.* **2018**, *21*, 88–145. [CrossRef]
10. Kalør, A.E.; Guillaume, R.; Nielsen, J.J.; Mueller, A.; Popovski, P. Network slicing in industry 4.0 applications: Abstraction methods and end-to-end analysis. *IEEE Trans. Ind. Inform.* **2018**, *14*, 5419–5427. [CrossRef]
11. Tan, W.; Wu, B.; Wang, S.; Huang, T. Large-scale Deterministic Transmission among IEEE 802.1 Qbv Time-Sensitive Networks. In Proceedings of the ICC 2022-IEEE International Conference on Communications, Seoul, Republic of Korea, 16–20 May 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 2315–2320.
12. Kreutz, D.; Ramos, F.M.; Verissimo, P.E.; Rothenberg, C.E.; Azodolmolky, S.; Uhlig, S. Software-defined networking: A comprehensive survey. *Proc. IEEE* **2014**, *103*, 14–76. [CrossRef]
13. Bello, L.L.; Lombardo, A.; Milardo, S.; Patti, G.; Reno, M. Experimental assessments and analysis of an SDN framework to integrate mobility management in industrial wireless sensor networks. *IEEE Trans. Ind. Inform.* **2020**, *16*, 5586–5595. [CrossRef]
14. Wang, C.; Zhang, L.; Li, Z.; Jiang, C. SDCoR: Software defined cognitive routing for internet of vehicles. *IEEE Internet Things J.* **2018**, *5*, 3513–3520. [CrossRef]

15. Sun, S.; Gong, L.; Rong, B.; Lu, K. An intelligent SDN framework for 5G heterogeneous networks. *IEEE Commun. Mag.* **2015**, *53*, 142–147. [CrossRef]
16. Craciunas, S.S.; Oliver, R.S.; Chmelík, M.; Steiner, W. Scheduling real-time communication in IEEE 802.1 Qbv time sensitive networks. In Proceedings of the 24th International Conference on Real-Time Networks and Systems, Brest, France, 19–21 October 2016; pp. 183–192.
17. Pop, P.; Raagaard, M.L.; Craciunas, S.S.; Steiner, W. Design optimisation of cyber-physical distributed systems using IEEE time-sensitive networks. *IET Cyber-Phys. Syst. Theory Appl.* **2016**, *1*, 86–94. [CrossRef]
18. Dürr, F.; Nayak, N.G. No-wait packet scheduling for IEEE time-sensitive networks (TSN). In Proceedings of the 24th International Conference on Real-Time Networks and Systems, Brest, France, 19–21 October 2016; pp. 203–212.
19. Vlk, M.; Hanzálek, Z.; Brejchová, K.; Tang, S.; Bhattacharjee, S.; Fu, S. Enhancing schedulability and throughput of time-triggered traffic in IEEE 802.1 Qbv time-sensitive networks. *IEEE Trans. Commun.* **2020**, *68*, 7023–7038. [CrossRef]
20. Nasrallah, A.; Thyagaturu, A.S.; Alharbi, Z.; Wang, C.; Shao, X.; Reisslein, M.; Elbakoury, H. Performance comparison of IEEE 802.1 TSN time aware shaper (TAS) and asynchronous traffic shaper (ATS). *IEEE Access* **2019**, *7*, 44165–44181. [CrossRef]
21. Zhao, L.; Pop, P.; Gong, Z.; Fang, B. Improving latency analysis for flexible window-based GCL scheduling in TSN networks by integration of consecutive nodes offsets. *IEEE Internet Things J.* **2020**, *8*, 5574–5584. [CrossRef]
22. Gavriluț, V.; Pop, P. Scheduling in time sensitive networks (TSN) for mixed-criticality industrial applications. In Proceedings of the 2018 14th IEEE International Workshop on Factory Communication Systems (WFCS), Imperia, Italy, 13–15 June 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–4.
23. Böhm, M.; Wermser, D. Multi-domain time-sensitive networks—Control plane mechanisms for dynamic inter-domain stream configuration. *Electronics* **2021**, *10*, 2477. [CrossRef]
24. Leonardi, L.; Bello, L.L.; Patti, G. Exploiting Software-Defined Networking to improve runtime reconfigurability of TSN-based networks. In Proceedings of the 2022 IEEE 27th International Conference on Emerging Technologies and Factory Automation (ETFA), Stuttgart, Germany, 6–9 September 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 1–4.
25. Peng, G.; Wang, S.; Huang, Y.; Huo, R.; Huang, T.; Liu, Y. Traffic shaping at the edge: Enabling bounded latency for large-scale deterministic networks. In Proceedings of the 2021 IEEE International Conference on Communications Workshops (ICC Workshops), Montreal, QC, Canada, 14–23 June 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–6.
26. Krolikowski, J.; Martin, S.; Medagliani, P.; Leguay, J.; Chen, S.; Chang, X.; Geng, X. Joint routing and scheduling for large-scale deterministic IP networks. *Comput. Commun.* **2021**, *165*, 33–42. [CrossRef]
27. Tian, W.; Gu, C.; Guo, M.; He, S.; Kang, J.; Niyato, D.; Chen, J. Large-Scale Deterministic Networks: Architecture, Enabling Technologies, Case Study and Future Directions. *IEEE Netw.* **2024**. [CrossRef]
28. Huang, Y.; Wang, S.; Huang, T.; Liu, Y. Cycle-based time-sensitive and deterministic networks: Architecture, challenges, and open issues. *IEEE Commun. Mag.* **2022**, *60*, 81–87. [CrossRef]
29. Zhong, X.; Zhu, J.; Guo, B.; Li, Q.; Huang, S. An SDN-enabled Optical Transport Network Simulation Platform for Cross-domain TSN Service. In Proceedings of the 2021 Asia Communications and Photonics Conference (ACP), Shanghai, China, 24–27 October 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–3.
30. *IEEE Std 802-2014 (Revision to IEEE Std 802-2001)*; IEEE Standard for Local and Metropolitan Area Networks: Overview and Architecture. IEEE: New York, NY, USA, 2014; pp. 1–74.
31. Cavalcanti, D.; Perez-Ramirez, J.; Rashid, M.M.; Fang, J.; Galeev, M.; Stanton, K.B. Extending accurate time distribution and timeliness capabilities over the air to enable future wireless industrial automation systems. *Proc. IEEE* **2019**, *107*, 1132–1152. [CrossRef]
32. Seijo, Ó.; Iturbe, X.; Val, I. Tackling the Challenges of the Integration of Wired and Wireless TSN with a Technology Proof-of-Concept. *IEEE Trans. Ind. Inform.* **2021**, *18*, 7361–7372. [CrossRef]
33. Miranda, G.; Municio, E.; Haxhibeqiri, J.; Hoebeke, J.; Moerman, I.; Marquez-Barja, J.M. Enabling Time-Sensitive Network Management Over Multi-Domain Wired/Wi-Fi Networks. *IEEE Trans. Netw. Serv. Manag.* **2023**, *20*, 2386–2399. [CrossRef]
34. Wang, X.; Yao, H.; Mai, T.; Guo, S.; Liu, Y. Reinforcement Learning-Based Particle Swarm Optimization for End-to-End Traffic Scheduling in TSN-5G Networks. *IEEE/ACM Trans. Netw.* **2023**, *31*, 3254–3268. [CrossRef]
35. Atiq, M.K.; Muzaffar, R.; Seijo, Ó.; Val, I.; Bernhard, H.P. When IEEE 802.11 and 5G meet time-sensitive networking. *IEEE Open J. Ind. Electron. Soc.* **2021**, *3*, 14–36. [CrossRef]
36. Larrañaga, A.; Lucas-Estañ, M.C.; Martinez, I.; Val, I.; Gozalvez, J. Analysis of 5G-TSN integration to support industry 4.0. In Proceedings of the 2020 25th IEEE International conference on emerging technologies and factory automation (ETFA), Vienna, Austria, 8–11 September 2020; IEEE: Piscataway, NJ, USA, 2020; Volume 1, pp. 1111–1114.
37. *IEEE Std 802.1 Qcc-2018 (Amendment to IEEE Std 802.1 Q-2018 as amended by IEEE Std 802.1 Qcp-2018)*; IEEE Standard for Local and Metropolitan Area Networks–Bridges and Bridged Networks–Amendment 31: Stream Reservation Protocol (SRP'18) Enhancements and Performance Improvements. IEEE: Piscataway, NJ, USA, 2018; Volume 2018, pp. 1–208.
38. Varga, B.; Farkas, J.; Fejes, F.; Ansari, J.; Moldován, I.; Máté, M. Robustness and Reliability Provided by Deterministic Packet Networks (TSN and DetNet). *IEEE Trans. Netw. Serv. Manag.* **2023**, *20*, 2309–2318. [CrossRef]
39. Benzaoui, N.; Gonzalez, M.S.; Estarán, J.M.; Mardoyan, H.; Lautenschlaeger, W.; Gebhard, U.; Dembeck, L.; Bigo, S.; Pointurier, Y. Deterministic dynamic networks (DDN). *J. Light. Technol.* **2019**, *37*, 3465–3474. [CrossRef]

40. Wang, S.; Wu, B.; Zhang, C.; Huang, Y.; Huang, T.; Liu, Y. Large-scale deterministic IP networks on CENI. In Proceedings of the IEEE INFOCOM 2021-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Vancouver, BC, Canada, 10–13 May 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–6.
41. Nayak, N.G.; Dürr, F.; Rothermel, K. Incremental flow scheduling and routing in time-sensitive software-defined networks. *IEEE Trans. Ind. Inform.* **2017**, *14*, 2066–2075. [CrossRef]
42. Nayak, N.G.; Dürr, F.; Rothermel, K. Time-sensitive software-defined network (TSSDN) for real-time applications. In Proceedings of the 24th International Conference on Real-Time Networks and Systems, Brest, France, 19–21 October 2016; pp. 193–202.
43. Li, H.; Shou, G.; Hu, Y.; Liu, Y. SDN/NFV enhanced time synchronization in packet networks. *IEEE Syst. J.* **2020**, *15*, 5634–5645. [CrossRef]
44. Kim, M.; Hyeon, D.; Paek, J. ETAS: Enhanced time-aware shaper for supporting nonisochronous emergency traffic in time-sensitive networks. *IEEE Internet Things J.* **2021**, *9*, 10480–10491. [CrossRef]
45. Pang, Z.; Huang, X.; Li, Z.; Zhang, S.; Xu, Y.; Wan, H.; Zhao, X. Flow scheduling for conflict-free network updates in time-sensitive software-defined networks. *IEEE Trans. Ind. Inform.* **2020**, *17*, 1668–1678. [CrossRef]
46. IEEE 802.1 Working Group. IEC/IEEE 60802 TSN Profile for Industrial Automation. Available online: https://1.ieee802.org/tsn/iec-ieee-60802/ (accessed on 6 March 2024).
47. Guo, M.; Shou, G.; Xue, J.; Hu, Y.; Liu, Y.; Guo, Z. Cross-domain Interconnection with Time Synchronization in Software-defined Time-Sensitive Networks. In Proceedings of the Asia Communications and Photonics Conference, Beijing, China, 24–27 October 2020; Optica Publishing Group: Washington, DC, USA, 2020.
48. Zhang, Y.; Xu, Q.; Xu, L.; Chen, C.; Guan, X. Efficient flow scheduling for industrial time-sensitive networking: A divisibility theory-based method. *IEEE Trans. Ind. Inform.* **2022**, *18*, 9312–9323. [CrossRef]