IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 13 May 2024, accepted 26 May 2024, date of publication 3 June 2024, date of current version 10 June 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3408923

RESEARCH ARTICLE

Enhancing Reliability of Time-Triggered Traffic in Joint Scheduling and Routing Optimization Within Time-Sensitive Networks

BILAL OMAR AKRAM[1,2], NOR KAMARIAH NOORDIN[1,2], (Senior Member, IEEE), FAZIRULHISYAM HASHIM[1,2], (Member, IEEE), MOHD FADLEE A. RASID[1,2], MUSTAFA ISMAEL SALMAN[3], AND ABDULRAHMAN M. ABDULGHANI[4], (Member, IEEE)

[1]Department of Computer and Communication Systems Engineering, Faculty of Engineering, University Putra Malaysia (UPM), Serdang, Selangor 43400, Malaysia
[2]Wireless and Photonics Networks Research Centre of Excellence (WiPNET), Faculty of Engineering, Universiti Putra Malaysia (UPM), Serdang, Selangor 43400, Malaysia
[3]Department of Computer Engineering, College of Engineering, University of Baghdad, Baghdad 10071, Iraq
[4]Department of Communication Technology and Network, Faculty of Computer Science and Information Technology, University Putra Malaysia (UPM), Serdang, Selangor 43400, Malaysia

Corresponding authors: Mohd Fadlee A. Rasid (fadlee@upm.edu.my) and Nor Kamariah Noordin (nknordin@upm.edu.my)

This work was supported by Universiti Putra Malaysia (UPM).

ABSTRACT The development of new technological applications is driving the need for deterministic network communication that exhibits reliable behavior. This is particularly relevant for applications like automation and autonomous vehicles, which require real-time network responsiveness. This has led to the development of various technologies, including Time-Sensitive Networking (TSN). Recent studies have explored the optimization of scheduling and routing in TSN, as this is crucial for meeting TSN's objectives. Current methods primarily focus on scheduling Time-Triggered (TT) transmissions, often overlooking the scheduling of lower-priority traffic types. Many of these methods do not optimize routing, and only a few consider reliability. In this paper, we present the Optimized Hybrid Deterministic Scheduling and Routing Plus (OHDSR+) approach. This method prioritizes communications, optimizing both scheduling and routing of TT communications, while also addressing the requirements of lower-priority best-effort (BE) communications, and ensuring the reliability of TT communications. We evaluated our approach using various problem instances. The results demonstrate our approach's effectiveness in managing different traffic types and ensuring reliable transmission for TT traffic. Our model achieves significant improvements compared to the state-of-the-art REU-CP-R method, with reductions in total latency of up to 1.03%, total response time of up to 34.70%, and scheduling response time of up to 63.97%. Notably, OHDSR+ exhibits superior scalability, successfully finding solutions for all tested network sizes with a 100% success rate, while REU-CP-R achieved only an 87.5% success rate. This highlights OHDSR+'s ability to provide optimal solutions for networks of various sizes with minimal latency and response times.

INDEX TERMS Best-effort (BE) traffic, constraints programming (CP), joint scheduling and routing, redundancy, real-time communication, reliability, time-sensitive networking (TSN), time-triggered (TT) traffic.

I. INTRODUCTION

The rise of real-time-based applications has led to increase the need for technologies capable of providing both deterministic and reliable communication in their networks. Deterministic delivery refers to the guaranteed arrival of data in a predictable manner, while reliability ensures that this data is delivered without loss. Addressing these needs, a variety of technologies have emerged, among which Time-Sensitive Networking (TSN) stands out as one of the most sophisticated

The associate editor coordinating the review of this manuscript and approving it for publication was Cong Pu.

and recent advancements. TSN has gained recognition as a key solution for real-time network requirements in fields like avionics, industrial automation, and autonomous vehicles, offering both high levels of determinism and reliability.

As a leading standard established by the Institute of Electrical and Electronics Engineers Standards Association (IEEE SA) [1], Ethernet is recognized as one of the most common standards in the field of general communication networks, serving as a fundamental technology in networking infrastructure. Despite its widespread use, Ethernet is not essentially equipped to handle network environments demanding precise timing and high reliability [2]. This gap has led to the development of various specialized protocols, such as Avionics Full-Duplex Switched Ethernet (AFDX) for the aviation sector, EtherCAT, PROFINET, and Sercos III for industrial automation, along with CAN and FlexRay for autonomous vehicle systems. However, these protocols encounter challenges in terms of adapting to diverse industrial needs and achieving compatibility with the Ethernet framework [3].

Following the dissolution of the Audio Video Bridging Task Group [4], the IEEE 802.1 Working Group [5] formed a new entity known as the Time-Sensitive Networking Task Group (TSN TG) [6]. This initiative, launched in 2012 by the IEEE SA, focused on adapting Ethernet networks to meet real-time communication requirements. The TSN TG set out to create a series of standards, leading to the development of what is now recognized as the TSN technology. These standards aim to enhance real-time networks, specifically in areas such as latency, reliability, management, and synchronization [7], [8]. With its comprehensive suite of over 60 individual ISO/OSI layer 2 IEEE standards, TSN is often described as a comprehensive toolkit for IEEE802 networks [9].

The TSN framework categorizes network traffic into three tiers based on priority: Time-Triggered (TT) traffic, which holds the highest priority; Audio/Video Bridging (AVB) traffic, with a comparatively lower priority; and Best-Effort (BE) traffic, which is assigned the lowest priority [10]. Despite its top-tier status, TT traffic can still experience delays due to potential congestion caused by lower-priority traffic, potentially impacting its latency [11]. TT traffic demands bounded, reliable latency with no jitter. The IEEE 802.1Qbv standard [12], addresses this need through the implementation of Time-Aware Shaping (TAS), a mechanism that organizes the flow of TT traffic based on a pre-determined Gate Control List (GCL) [13]. For AVB traffic, which is of a lower priority, the Credit-Based Shaping (CBS) technique is utilized, as outlined in the IEEE 802.1Qav standard [14]. BE traffic, being the lowest in priority, does not require stringent timing. To ensure the smooth functioning of the TSN network, each component, whether a bridge (BR) or an end-system (ES), needs to be in sync. The design of the Gate Control Lists (GCLs) is crucial for the efficient handling of high-priority TT traffic while accommodating the needs of lower-priority traffic [15].

Once a transmission begins, regardless of its priority, it should be completed. To facilitate this, the IEEE 802.1Qbu standard [16] was introduced, which includes a frame preemption mechanism. This mechanism prioritizes high-priority traffic in a network by allowing it to interrupt an ongoing low-priority transmission. The high-priority transmission takes precedence, and once it is complete, the low-priority transmission resumes from the point of interruption.

Typical transmission protocols lack specific protection against transmission errors. To address this, TSN implemented the IEEE 802.1CB standard [17], which introduces the Frame Replication and Elimination for Reliability (FRER) mechanism. In terms of ensuring the reliability of traffic transmission, FRER is regarded as the sole effective mechanism put forward by any of the TSN standards [18].

Time synchronization across all network devices and the synchronization of data streams are crucial for real-time systems. The IEEE 802.1AS standard [19] tackles the issues of synchronization and timing, aiding in the prevention of data loss due to collisions and enhancing the timeliness of network traffic. This standard employs a global time protocol to ensure synchronization across all network devices.

Our main contribution is the development of a model named Optimized Hybrid Deterministic Scheduling and Routing Plus (OHDSR+). This model is an enhanced version of our earlier model, OHDSR [20], and it focuses on managing the routing and scheduling of Time-Triggered (TT) traffic, while also taking into account Best-Effort (BE) traffic based on their respective priorities. The OHDSR+ model particularly addresses the reliability of TT communications in both routing and scheduling by ensuring that traffic has a redundant copy using an alternate route, and it also highlights the time shift between the original and redundant traffic. The contributions of our paper can be summarized as follows:

- We developed the OHDSR+ model as an enhanced version of the Optimized Hybrid Deterministic Scheduling and Routing (OHDSR) framework.
- OHDSR+ manages the routing and scheduling of both Time-Triggered (TT) and Best-Effort (BE) traffic, considering their respective priorities using the Queue-Gating Time (QGT) approach.
- Enhanced Reliability for TT Traffic: OHDSR+ prioritizes the reliability of TT communications through: (1) Redundant Routing: Ensuring traffic has a redundant copy transmitted via an alternate route. (2) Time Shift Implementation: Implementing a temporal shift between the original and redundant traffic to further enhance reliability and minimize data loss.

The structure of this paper is as follows: Section II provides a review of related work. Section III outlines the system models. In Section IV, we introduce our optimization framework, elaborating on the constraints and objective functions. The outcomes of our evaluations are discussed in Section V. Finally, the paper concludes with Section VI.

II. RELATED WORKS

A variety of scheduling mechanisms have been introduced in the literature to identify the optimal schedule for TSN networks, taking into consideration various factors such as response time, scalability, latency, worst-case latency, and others. These proposed mechanisms often view scheduling as a primary factor, reflecting its significance in the standard scheme introduced by the TSN Task Group, particularly in the IEEE 802.1Qbv standard, which does not specify any particular scheduling approach. Additionally, many mechanisms include joint routing in their scope due to its impact on scheduling; more optimal routing contributes to a more effective time-sensitive schedule. Furthermore, several studies have also considered the reliability of network communications, acknowledging its crucial role in minimizing data loss and ensuring high-quality Quality of Service (QoS).

In the literature, various approaches have been applied to the problem of joint routing and scheduling. Many studies have utilized different heuristic algorithms due to their effectiveness in solving complex problems. However, it is important to note that heuristic approaches typically aim for a reasonable solution rather than an optimal one. For instance, Wang et al. [21] presented an approach based on Ant-Colony Optimization (ACO), termed Improved ACO (IACO), which demonstrated reasonable latency and jitter for network traffic. Pahlevan and Obermaisser [22] employed Genetic Algorithms (GA) in their approach, focusing on multicast streams. Their findings suggest that joint routing enhances scheduling possibilities compared to fixed routing, thereby improving the schedulability of their solution. In their subsequent work [23], they developed a Heuristic List Scheduler (HLS) and compared it to the basic List Scheduler (LS) approach, achieving a 28% improvement in make span and increased schedulability. Gavrilut et al. [24], [25] used a Greedy Randomized Adaptive Search Procedure-based (GRASP) algorithm, considering both AVB and TT traffic. Additionally, many researchers have integrated reliability into their heuristic-based work. For example, a study [26] introduced a fault-resilient algorithm that effectively balances reduced financial costs with feasible routing and scheduling for TT traffic. Another noteworthy heuristic approach [27] focuses on enhancing load balance on links and throughput, considering the reliability of redundant streams through frame replication and elimination functions.

Optimization methods, widely used to identify the best feasible or optimal solution through various constraints and specific objectives, have been extensively applied in joint routing and scheduling problems. Integer Linear Programming (ILP) is one such method, employed in [28], where factors like the number of streams — more impactful on response time than topology size — were considered. Some studies utilizing ILP, like Yu and Gu [29], claim their work increases schedulability and minimizes response time compared to other related works. Schweissguth et al. [30], [31] are noted for being among the early works considering multicast streams in their ILP-based approach. Another optimization method, Satisfiability Modulo Theories (SMT), was the basis of Xu et al. [32] approach, which enhanced scalability in large-scale networks while reducing complexity.

Beyond heuristic and optimization methods, some recent works, like Yang et al. [33], have started adopting machine learning to solve joint routing and scheduling problems, considering AVB and BE traffic alongside TT traffic. It is worth noting that task scheduling has been addressed from various perspectives. Notably, Ali et al. [34] and Tariq et al. [35] focused on energy-efficient task scheduling in real-time applications on heterogeneous Network-on-Chip (NoC) architectures.

Reliability has also been a key consideration in many works using optimization techniques. For instance, [36] addresses unexpected link failures in TT traffic, using an ILP solver to mitigate their impact on network scheduling. Similarly, [37] developed an ILP-based scheme for large-scale networks, computing multipath routing with reliability constraints and no-wait schedules. Balasubramanian et al. [38] proposed Fed-TSN, a novel framework that leverages federated learning to optimize fault recovery in TSN within industrial IoT settings. Their approach addresses the limitations of traditional protection mechanisms, particularly in scenarios with multiple correlated link failures. Fed-TSN encompasses three key components: the TSNu1 algorithm, which utilizes SDN for path protection in the case of independent failures; the TSNu2 algorithm, which employs heuristics to minimize the joint failure probability of primary and backup paths in scenarios with correlated failures; and a federated learning mechanism to intelligently predict optimal recovery paths and TSN gate schedules. Through simulation, the authors demonstrated that Fed-TSN achieves higher bandwidth utilization compared to benchmark approaches, particularly in the presence of multiple link failures.

Another study by Feng et al. [39] introduces ReT-FTS, a novel approach to enhancing the reliability of TSN by incorporating re-transmission mechanisms into the scheduling process. The researchers developed analytical techniques to evaluate network reliability under various conditions, including scenarios where specific AVB streams are intentionally dropped. Their findings indicate that the strategic dropping of streams does not necessarily improve overall reliability. To address this, a novel scheduling algorithm is proposed that balances the trade-off between reliable communication and efficient resource utilization within TSN. Extensive evaluations, utilizing both synthetic and realistic network scenarios, validate the efficacy of ReT-FTS in ensuring both the reliability and schedulability of TSN.

A novel method to enhance the reliability TSN by addressing limitations in the FRER mechanism introduced by Hu et al. [40]. This method incorporates a path reliability model that considers both link and node characteristics,

enabling the selection of highly reliable paths. Furthermore, the authors developed the Edge-Disjoint Path Pairs Selection (EDPPS) algorithm to identify two such paths for redundant data transmission, ensuring delivery even in the event of failures. Evaluations of the FRER-EDPPS approach demonstrate improved path reliability and reduced delay jitter, especially under high network loads, compared to the conventional FRER and FRER-MPC mechanisms. Min [41] investigated the complex interplay of multipath routing and TAS scheduling within the FRER mechanism TSN. The authors propose the Member Wait-and-Forward (MWF) technique to optimize resource utilization following frame elimination and address potential MWF-induced deadlock issues with a topological sorting solution. Further, they introduce a Redundancy-Weighted Multipath Routing (RWMR) algorithm to enhance transmission reliability, and a Last Arrival Time-based TAS Scheduler (LATS), augmented with metaheuristic optimization, to improve scheduling performance. Extensive evaluations highlight the efficacy of these strategies, demonstrating increased support for concurrent flows, reduced resource utilization, and enhanced overall schedulability, while effectively managing the complexities inherent in routing and scheduling within the FRER framework.

Constraint Programming (CP) is another popular method in this field. Vlk et al. [42] introduced two CP-based models aiming to improve schedulability in large-scale networks. A notable CP-based approach, considering reliability, was introduced in [43]. This method applied redundancy to streams and imposed a routing constraint to ensure different paths for redundant streams. In our previous work [20],we introduced a joint routing and scheduling approach based on CP, accommodating both TT and BE traffics and handling multicast streams. This paper will extend our previous work by incorporating considerations of reliability and urgent traffic.

III. SYSTEM MODEL AND PROBLEM DEFINITION

In this section, we will delve into our system model and examine its components for a comprehensive understanding. We will begin with an exhaustive description of the network model, followed by an in-depth look at the application model. Next, the bridge model will be described in detail. Finally, we will conclude this section by explaining the problem. For your convenience, Table 1 lists all the notations used in this discussion, along with their descriptions.

A. SYSTEM MODEL

In our research, the network topology is depicted as a directed graph $\mathcal{G} =< \mathcal{V}, \mathcal{E} >$, which includes nodes ($\mathcal{V}$) and edges ($\mathcal{E}$). This network comprises two varieties of nodes. The first category encompasses End-Systems (ES), which are devices that transmit (termed 'talkers') and receive (referred to as 'listeners') data. An end-system can simultaneously serve as both a talker and a listener for various data types, representing either users or servers within the network. The second type of node is the Bridges (BR), which are devices linking multiple network nodes together. The edges, denoted as (υ_a, υ_b), are the connections that link two distinct nodes. These connections might occur between end-systems and bridges or between two bridges. They facilitate bi-directional and full-duplex communication. The network functions in a synchronized manner, conforming to the Time-Sensitive Networking (TSN) standards, ensuring cohesive operation among all nodes.

Our system model, symbolized as a Directed Acyclic Graph (DAG), is composed of a set of applications, each denoted as $\delta_k \in \Delta$. Within each application, there is a collection of tasks representing the operations executed at the end-systems. Each task, identified as τ_j, is part of the overall task set T for all applications. Another crucial element of each application is the streams, labeled σ_i and belonging to the set S. These streams facilitate data communication between the end-systems and traverse the bridges by passing through the edges that connect the nodes. The functioning of these streams is closely linked to the operation of the tasks within the same application.

TABLE 1. Notations.

Symbol	Description
HP	Hyperperiod
$\mathcal{G} =< \mathcal{V}, \mathcal{E} >$	Network graph
$\upsilon_a \in \mathcal{V}$	Node
$es_m \in ES$	End-System
$br_n \in BR$	Bridge
$q_\phi^{br_n} \in Q^{br_n}$	Queue number ϕ of the bridge br_n
$\rho_\phi^{q^{br_n}}$	Queue type (TT or BE) of the bridge br_n
$t_\varphi^{br_n} \in \Phi^{br_n}$	Gating time number φ of the bridge br_n
$g_q^{br_n} \in G_q^{br_n}$	Gate of the queue q at the bridge br_n
$\psi_g^{br_n}$	Gate status (1 or 0) of the queue g at the bridge br_n
$\kappa_\varphi^{q^{br_n}}\boxed{\psi}$	Queue-Gating Time (QGT) for each queue q at status ψ
$(\upsilon_a, \upsilon_b) \in \mathcal{E}$	Edge (Link)
$\Omega_{(\upsilon_a,\upsilon_b)}$	Link transmission speed
$\delta_k \in \Delta$	Application
Γ_k^δ	Application period
$\tau_j \in T$	Task
es_j^τ	Execution end-system
ω_j^τ	Worst-case execution time
Γ_j^τ	Task period
$\eta_{\upsilon_a}^\tau$	Task offset at node υ_a
$\zeta_{\upsilon_a}^\tau$	Task end-time at node υ_a
$\sigma_i \in S$	Stream
L_i^σ	Stream size
Γ_i^σ	Stream period
ρ_i^σ	Stream type (TT or BE)
Π_i^σ	Stream redundancy status
$\eta_{(\upsilon_a,\upsilon_b)}^\sigma$	Stream offset at link (υ_a, υ_b)
$\varepsilon_{(\upsilon_a,\upsilon_b)}^\sigma$	Stream duration at link (υ_a, υ_b)
$\zeta_{(\upsilon_a,\upsilon_b)}^\sigma$	Stream end-time at link (υ_a, υ_b)
$\tau_{talker}^{\sigma_i}$	Source task
$T_{listener}^{\sigma_i}$	Destination tasks
$\mathbb{R}_i^\sigma$	Stream route

Applications are characterized by their periods, with each application having its unique period Γ_k^{δ}. The Hyperperiod (*HP*), which is the time span over which the network communication execution repeats, is determined by the least common multiple (lcm) of these application periods. In our discussion, we primarily use microseconds (μs) and milliseconds (ms) as our time units.

Each task is characterized by its own period, Γ_j^{τ}, indicating the duration of its execution. The Worst-Case Execution Time (WCET) for a task is denoted as ω_j^{τ}. Additionally, the end-system where a task is executed is indicated as es_j^{τ}. Tasks can be categorized as either talker end-system tasks or listener end-system tasks, meaning they are executed at either the sending or receiving device. These tasks are associated with the timing of the corresponding streams, with the talker task's end-time labeled as $\tau_{talker}^{\sigma_i}$ being crucial for initiating the data stream transmission.

Streams in our model facilitate communication between two tasks and can be either unicast or multicast, potentially reaching multiple destinations with listener tasks $T_{listner}^{\sigma_i}$. Listener tasks initiate after the receiving end-systems get the incoming streams. Similar to tasks, each stream has its own period, Γ_i^{σ}, within which it should be received by the destination end-systems. These streams also have a size constraint, where they must be less than or equal to 1500 bytes, aligning with the Maximum Transmission Unit (MTU) limit.

The TT streams in our system can either be redundant, traversing two separate routers, or non-redundant, passing through a single route. In contrast, BE streams are always transferred along a single route. The redundancy status of each stream is indicated by the parameter Π_i^{σ}. When $\Pi_i^{\sigma} = 1$, it signifies that the stream is redundant and takes an alternative path. Conversely, when $\Pi_i^{\sigma} = 0$, it indicates that the stream is not redundant, lacking any alternative routing path. Streams traversing from one end-system to another, navigate through one or more bridges. These bridges are specifically designed to function in networks that exhibit Time-Sensitive Networking (TSN) characteristics. Each bridge, denoted as $br_n \in BR$, plays a role in managing and directing different types of traffic streams towards the subsequent node, prioritizing them accordingly.

In our model, all bridges share a uniform design, featuring eight egress ports. Each of these ports is equipped with a queue, designated to temporarily store the streams. These queues, represented as $q_{\phi}^{br_n} \in Q^{br_n}$ are differentiated by their types ρ_{ϕ}^{q}, which are determined based on the type of stream ρ_i^{σ} stored within.

When allocating streams to queues, if multiple queues share the same type, the new stream is assigned to the queue with the least accumulation. In cases where all relevant queues have an equal number of streams, the new stream is placed in the first-order queue. To enhance clarity in our documentation, after the initial mention of a bridge-specific notation (such as br_n), subsequent references will omit the bridge identifier. For example, the first queue will be referred to simply as q_0 instead of $q_0^{br_n}$, starting from its second mention. This approach is adopted for clearer illustration and ease of understanding.

Each queue is paired with a gate, denoted as $g_q \in G_q$, that regulates the status of the egress ports and controls the flow of traffic streams exiting the queue. The state of these gates, symbolized as ψ_g, is governed by a predefined set known as Gate Control Lists (GCLs). The gates can be either closed, which is their default state indicated by zeros "0", or open, represented by ones "1".

The GCL features an eight-digit binary entry that reflects the status of all gates, with each digit corresponding to the gate status of one of the eight queues. These digits are arranged from right to left in order of the queues. The entries in the GCL are dynamic, altering according to the gate statuses, with the moments of these changes termed as gating times, denoted as $t_{\varphi} \in \Phi$.

As mentioned earlier, queues are categorized into two types based on the stream type they accommodate. This leads to the classification of Time-Triggered (TT) and Best-Effort (BE) queues. In our model, the first two queues (q_0 and q_1) are designated as TT queues, indicated by $\rho_0^q = \rho_1^q = TT$, while the last two queues (q_6 and q_7) are classified as BE queues, noted as $\rho_6^q = \rho_7^q = BE$. The remaining queues are idle, not assigned to any specific traffic type, and are tagged as "N/U", signifying that they are not used in the current setup.

In this paper, we will utilize the Queue-Gating Time (QGT), denoted as $\kappa_{\varphi}^{q}\boxed{\psi}$, to control gate statuses at each specified gating time. This concept was thoroughly discussed in our prior work [20], but a concise explanation will be provided here. We consider two gating times: $t_1 = 0\mu s$ and $t_2 = 500\mu s$, with the gate status entry "11000011" designated at t_1 and "00000000" at t_2. This indicates that the gates for the first two and the last two queues will be open at $0\mu s$, and all gates will close at $500\mu s$. As an example, for queue q_0 at gating time $t_1 = 0\mu s$, which has an open gate status $\psi_0 = 1$, the QGT is determined as $\kappa_1^0\boxed{1} = 0\mu s$.

B. PROBLEM DEFINITION

Scheduling is a critical aspect of Time-Sensitive Networking (TSN), as strongly emphasized in its core standards. The optimal scheduling of data streams and processing tasks is essential for real-time systems, particularly those that demand low latency and zero data loss. Optimal routing for time-sensitive data streams plays a significant role in enhancing scheduling efficiency, thereby further reducing latency in routed streams. Reliability, a key focus of TSN standards, is essential for ensuring dependable system communications and preventing data loss. Together, an optimized and reliable schedule, paired with an efficient routing strategy within the system model, contributes significantly to achieving the goals of TSN, such as seamless real-time communication and enhanced data integrity.

TSN systems typically consist of multiple end-systems and bridges, interconnected by various links. Tasks executed on these end-systems generate data streams, which are then routed through the bridges from one end-system to another. In every TSN system model, the reliable routing of these streams is crucial, as is the effective scheduling of tasks and streams. Our primary objective is to achieve this with minimal response time and latency, while simultaneously ensuring the reliability of the routed streams. This dual focus on speed and reliability is fundamental to the effective functioning of any TSN system.

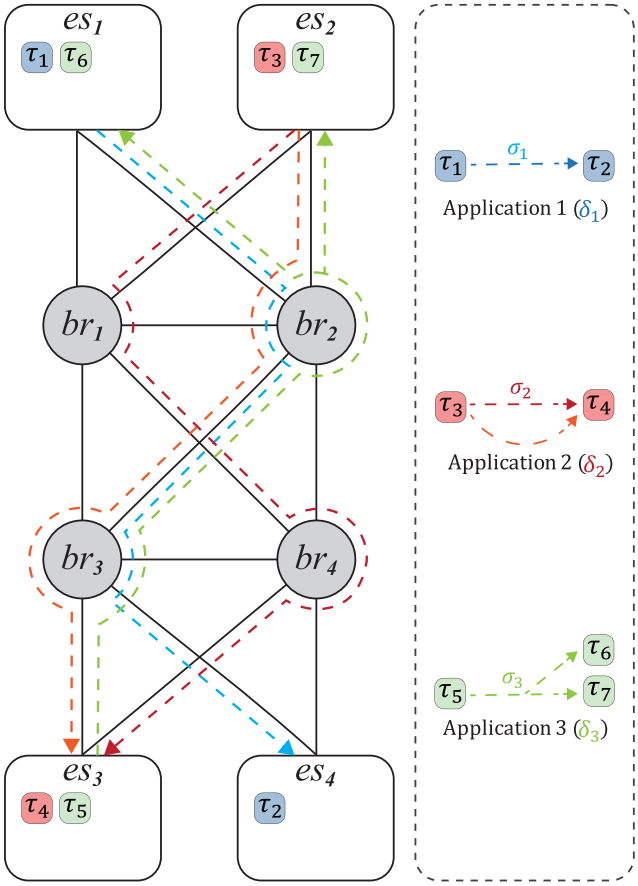


FIGURE 1. The network topology of the exemplary problem instance illustrates the route taken by each stream.

C. EXEMPLARY PROBLEM INSTANCE

Our exemplary instance is designed to illustrate the system model of our approach. As shown in Fig 1, our network topology includes four end-systems interconnected with four bridges via links, each operating at a speed of $\Omega_{(\upsilon_a,\upsilon_b)} = 10\text{Mbps}$. The network hosts three applications, denoted as $\Delta = \{\delta_1, \delta_2, \delta_3$, each comprising a single stream and a varying number of tasks. Specifically, the first two applications consist of two tasks each, while the third application includes three tasks. Each stream, $\sigma_i \in S$, is defined by a size L_i^σ; the first two streams, categorized as Time-Triggered (TT) streams, are each $65Bytes$, while the third stream, a Best Effort (BE) stream, is $35Bytes$. The first and third streams are designated as non-redundant, with their stream redundancy status being $\Pi_i^\sigma = 0$. This means that these streams, both the first and the third, do not take alternate paths and are routed through a single route. On the other hand, the second stream is set as redundant, indicated by a stream redundancy status of $\Pi_i^\sigma = 1$. Consequently, this stream possesses an alternative stream that follows a different path, offering an additional layer of reliable routing. This increased redundancy enhances the overall reliability of the stream, ensuring better robustness in the network.

The tasks, $\tau_j \in T$, are executed on the end-systems (ESs). These occur either at the talker ES, where they generate streams upon completion, or at the listener ES, where tasks are executed upon receiving the corresponding streams. The Worst-Case Execution Time (WCET) ω_j^τ for tasks in the first two applications is set at $35\mu s$, while for the third application, it is $25\mu s$. We have synchronized the deadlines of the streams and tasks with their respective periods, each fixed at $1000\mu s$, ensuring consistency across tasks and streams within the applications. Consequently, the Hyperperiod (HP) for our applications is determined as $HP = \text{lcm}\{\delta_1, \delta_2, \delta_3\} = \text{lcm}\{1000, 1000, 1000\} = 1000\mu s$. Table 2 provides a detailed overview of the parameters related to the applications, streams, and tasks within our exemplary problem instance.

TABLE 2. The applications, streams, and task parameters of the exemplary problem instance.

Δ	Γ_k^δ	T	ω_j^τ	Γ_j^τ	S	ρ_i^σ	L_i^σ	Γ_i^σ	Π_i^σ
δ_1	1000	τ_1, τ_2	35	1000	σ_1	TT	65	1000	0
δ_2	1000	τ_3, τ_4	35	1000	σ_2	TT	65	1000	1
δ_3	1000	τ_5, τ_6, τ_7	25	1000	σ_3	BE	35	1000	0

Each bridge in our network configuration is equipped with eight queues. The first two are allocated for TT streams, the last two for BE streams, while the remaining four middle queues are presently unused, as depicted in Fig 2.

Aligned with the established period time, we have defined four gating times to signify moments when the gate statuses undergo changes. These changes, as elaborated in Fig 2, correspond to four distinct gate status modifications in accordance with the Gate Control List (GCL) Entry. The initial change takes place at $0\mu s$, where the Time-Triggered (TT) gates g_0 and g_1 are opened ($\psi_0 = \psi_1 = 1$), while all other gates remain closed. Subsequently, at $450\mu s$, there is a transition marking the closure of TT gates ($\psi_0 = \psi_1 = 0$).

The third and fourth alterations are related to the Best Effort (BE) gates g_6 and g_7. These gates are opened ($\psi_6 = \psi_7 = 1$) at $650\mu s$ and then closed ($\psi_6 = \psi_7 = 0$) at $1000\mu s$. In line with the GCL entry detailed in Fig 2, the Queue-Gating Time (QGT) $\kappa_\varphi^q[\psi]$ is initially set to "$0\mu s$ " for all eight queues at the first gating time $t_1 = 0\mu s$. For the subsequent second, third, and fourth gating times, the QGT for all queues

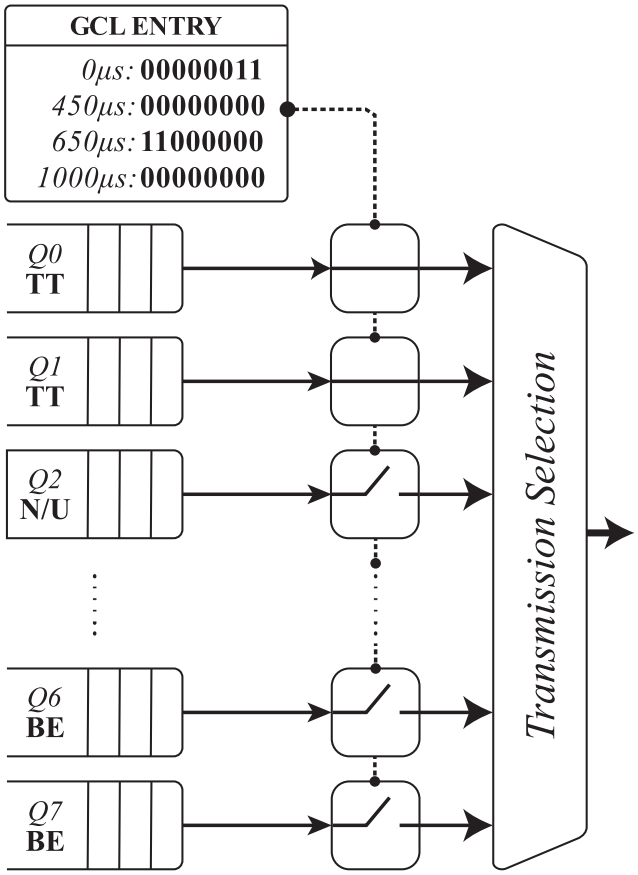


FIGURE 2. A simplified TSN bridge in the exemplary problem instance features queues and gates, along with a Gate Control List (GCL) entry.

is assigned as $\kappa_2^q[\psi] = t_2 = 450\mu s$, $\kappa_3^q[\psi] = t_3 = 650\mu s$, and $\kappa_4^q[\psi] = t_4 = 1000\mu s$, respectively.

The transmission schedule for our exemplary problem instance is detailed in Fig 3. The first application features a TT transmission with a unicast stream, as depicted in Fig 3(a). Following this, we have the transmission of Application 2, which involves a TT stream with a stream redundancy status of $\Pi_i^\sigma = 1$. This indicates that the stream must be redundantly routed through two different paths, with a temporal shift between the original stream σ_{2A} and its redundant counterpart σ_{2B}. As shown in Fig 3(b), σ_{2B} follows a distinct route from σ_{2A} and commences transmission after σ_{2A}. This variation in path and transmission start time enhances the reliability of the streams and reduces the probability of data loss. Upon the arrival of either the original or the redundant stream at the listener end-system, the listener task is initiated, and the other stream is disregarded, as the data has already been received. Fig 3(c) illustrates the schedule of Application 3, which features a multicast BE stream transmitting from one talker end-system to two distinct listener end-systems.

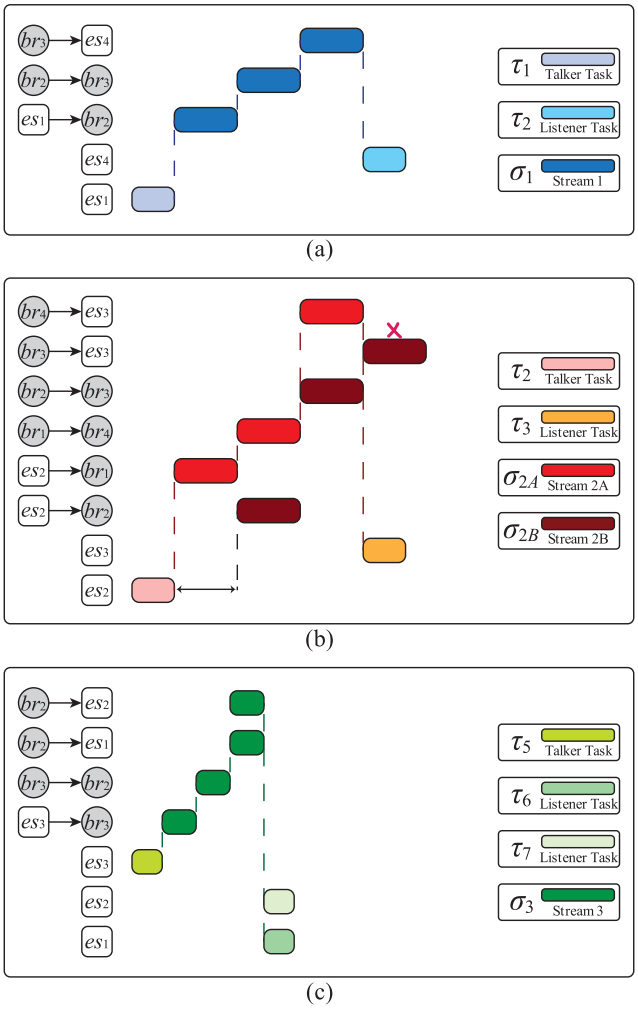


FIGURE 3. The network schedule for the exemplary problem instance includes: (a) transmission of Application 1 (δ_1) for a unicast time-triggered (TT) stream, (b) transmission of Application 2 (δ_2) for a unicast redundant TT stream, and (c) transmission of Application 3 (δ_3) for a multicast best-effort (BE) stream.

IV. OPTIMIZATION FRAMEWORK

This section presents our approach to optimizing scheduling and routing while incorporating the reliability of network transmissions. Identifying the optimal route and schedule in time-sensitive networks is typically classified as an NP-hard problem, given the increasing complexity of these challenges in proportion to network complexity. In our framework, we employ Constraint Programming (CP) as the paradigm to address these optimization challenges. This approach represents a development of our previous work, "Optimized Hybrid Deterministic Scheduling and Routing" (OHDSR) [20]. In this paper, we introduce an enhanced version of this approach, designated as OHDSR+. The OHDSR+ model not only encompasses the scheduling and routing features of its predecessor but also integrates additional enhancements and focuses explicitly on the reliability aspects of the network. Figure 4 provides an overview of our OHDSR+ model.

A. CONSTRAINTS

Identifying the optimal solution for any optimization problem invariably involves navigating through a range of constraints

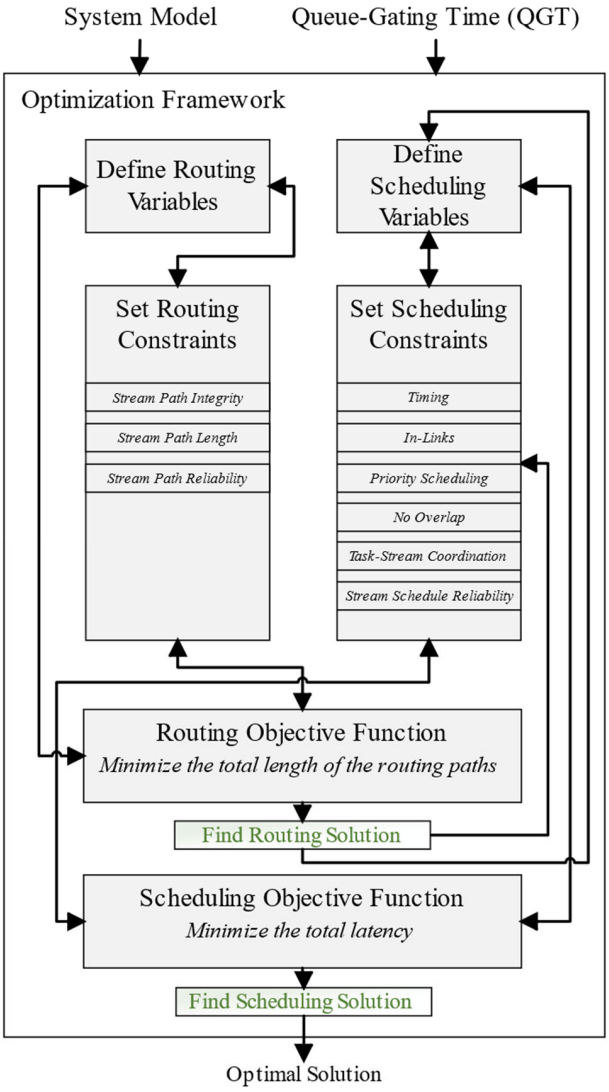


FIGURE 4. Overview of our OHDSR+ model.

that must be satisfied. Each optimization framework is characterized by its unique set of variables, representing the elemental components of the problem at hand. The inter-relationships among these variables are established through constraints, which function as limitations or boundaries governing the solution space. In this section, we will delve into the specific constraints employed in our approach, outlining their roles and how they shape the optimization process.

1) ROUTING CONSTRAINTS

The first category of constraints in our model pertains to the routing of streams. These constraints are a continuation of our previous work [20] and are influenced by the research presented in [43] and [44]. In our model, the route of each stream, denoted as σ_i, can be established by tracing a reverse path from the listener end-system back to the talker end-system. As the routed stream traverses the network, it passes through one or more intermediary nodes, in addition to the sender and receiver nodes. For each node υ_a on this path, we denote a possible successor node as $Z_{\upsilon_a}^{\sigma_i}$. The value assigned to this potential successor node varies based on its position and function: it is set to 'nil' if the node does not lie on the stream's path or is not a successor; it is equal to υ_a if the current node is the sender node; and it takes the value of υ_b for the direct successor of the sender node υ_a.

In order of find the route of each stream, the successor nodes need to be identified. The successor node of the node υ_a along the path of the stream σ_i, denoted as $Z_{\upsilon_a}^{\sigma_i}$. This can be established through a reverse route, beginning the procedure at the receiver nodes and progressively constructing it as a tree structure. The possible values of $Z_{\upsilon_a}^{\sigma_i}$ is: $\{\upsilon_a\} \cup \upsilon_b \in \mathcal{V}$ *where* $(\upsilon_a, \upsilon_b) \in E\} \cup \{nil\}$.

Referring to the exemplary instance in section III-C, let us consider one of the destinations of the multicast stream σ_3. The path for this stream is delineated as: $[es_3 \longrightarrow br_3 \longrightarrow br_2 \longrightarrow es_1]$. Consequently, the possible successor node $Z_{\upsilon_a}^{\sigma_3}$ is determined as follows: it will be es_3 if υ_a equals es_3, and br_3 if υ_a is br_3. Similarly, $Z_{\upsilon_a}^{\sigma_3}$ will equal br_3 if υ_a is br_2, and br_2 if υ_a is es_1. For any node υ_a that is not on the path of stream σ_3 (such as es_2, es_4, br_1, and br_4), $Z_{\upsilon_a}^{\sigma_3}$ will be assigned as "*nil*". Furthermore, the length of the path from node υ_a to the talker-node of stream σ_i is denoted as $\Upsilon_{\upsilon_a}^{\sigma_i}$, with its permissible range defined as $\{0 \leq \Upsilon_{\upsilon_a}^{\sigma_i}, \leq |BR| + 1\}$. The specific routing constraints relevant to this scenario are elaborated in equations 1 to 7, categorized into three groups.

a: STREAM PATH INTEGRITY CONSTRAINTS

The selection of a route path is contingent upon the relationships between each node and its adjacent predecessor and successor nodes. To define these relationships, stream path integrity constraints are articulated through four distinct equations. Given that our path selection process is conducted in reverse order, it begins with the listener end-system nodes. According to this approach, any listener end-system must be associated with a successor node, which is essentially the last bridge responsible for transmitting data to the listener. This principle is encapsulated in (1).

$$\forall \sigma_i \in S, \forall \upsilon_a \in es\langle \tau_{listener}^{\sigma_i} \rangle, \textit{where } \tau_{listener}^{\sigma_i} \in T_{listener}^{\sigma_i} : \quad Z_{\upsilon_a}^{\sigma_i} \neq nil \tag{1}$$

$$\forall \sigma_i \in S, \forall \upsilon_a \in es\langle \tau_{talker}^{\sigma_i} \rangle : \quad Z_{\upsilon_a}^{\sigma_i} = \upsilon_a \tag{2}$$

$$\forall \sigma_i \in S, \forall \upsilon_a \in ES \textit{ except } \left\{ es\langle \tau_{listener}^{\sigma_i} \rangle and es\langle \tau_{talker}^{\sigma_i} \rangle \right\}, \textit{where } \tau_{listener}^{\sigma_i} \in T_{listener}^{\sigma_i} : \quad Z_{\upsilon_a}^{\sigma_i} = nil \tag{3}$$

$$\forall \sigma_i \in S, \forall \upsilon_a, \upsilon_b \in V : \quad Z_{\upsilon_b}^{\sigma_i} = nil \iff Z_{\upsilon_a}^{\sigma_i} \neq \upsilon_b \tag{4}$$

The network is composed of a series of end-systems and bridges. According to our model, aside from the listener and talker end-systems, no other end-system will possess a

successor node. Notably, the talker end-system is defined to have itself as its successor. These concepts are detailed in (2) and (3). With the understanding that the listener end-system has a designated successor, it becomes necessary to impose constraints on the other nodes along the paths. Equation (4) addresses this by ensuring that each node with a successor must also have a predecessor. This constraint is crucial for maintaining the integrity and continuity of the path from the talker to the listener within the network.

b: STREAM PATH LENGTH CONSTRAINTS

In the context of reverse path selection, the talker end-system is assigned a predecessor node, which is typically the first bridge that forms part of the stream's path and connects to the talker end-system. As previously mentioned, each node along the path that has a predecessor must also have a successor. In the case of the talker end-system, it is designated as its own successor, as it represents the terminal node in the path. Consequently, the length of the path for the talker end-system is deemed to be zero, a concept that is further elucidated in (5).

$$\begin{aligned}&\forall \sigma_i \in S, \forall \upsilon_a \in es\langle \tau^{\sigma_i}_{talker}\rangle :\\ &\quad Z^{\sigma_i}_{\upsilon_a} = \upsilon_a,\\ &\quad \Upsilon^{\sigma_i}_{\upsilon_a} = 0 \end{aligned} \tag{5}$$

$$\begin{aligned}&\forall \sigma_i \in S, \forall \upsilon_a \in \mathcal{V}\ exceptes\langle \tau^{\sigma_i}_{talker}\rangle :\\ &\quad \left(Z^{\sigma_i}_{\upsilon_a} \neq nil\right) \Longrightarrow \left(\Upsilon^{\sigma_i}_{\upsilon_a} = \Upsilon_{Z^{\sigma_i}_{\upsilon_a}} + 1\right) \end{aligned} \tag{6}$$

Implementing an acyclic path constraint is crucial in any optimization model that involves routing. This can be achieved by ensuring that the length of any node along the path is exactly one more than the length of its successor node. This principle is fundamental to preventing cycles within the routing paths. The mathematical formulation of this constraint is demonstrated in (6).

c: STREAM PATH RELIABILITY CONSTRAINT

One of the most critical aspects of this work is the reliability of communications. To enhance the transmission reliability of certain TT streams across the network, redundancy will be applied. For each stream designated with a stream redundancy status of $\Pi^{\sigma}_{i} = 1$, we will enforce the constraint outlined in (7). This constraint ensures that for any stream where redundancy is applied, the redundant stream σ_{iB} will choose a path different from the original stream σ_{iA}. It achieves this by stipulating that the successor node for the redundant stream must not be the same as that of the original stream, a rule that applies to all nodes involved.

$$\begin{aligned}&\forall \sigma_i \in S, \Pi^{\sigma}_{i} = 1, \forall \upsilon_a \in \mathcal{V}\ except\ es\langle \tau^{\sigma_i}_{talker}\rangle :\\ &\quad Z^{\sigma_{iA}}_{\upsilon_a} \neq Z^{\sigma_{iB}}_{\upsilon_a} \end{aligned} \tag{7}$$

2) SCHEDULING CONSTRAINTS

Scheduling constraints are vital for ensuring the optimality of our framework. These constraints are designed to ensure that the scheduling of streams and tasks is conducted in a manner that aligns with the deterministic goals of Time-Sensitive Networking (TSN). In this section, we will delve into the specific scheduling constraints applicable to both streams and tasks within our model. To facilitate a clearer understanding and organization, we have categorized these constraints into six distinct groups, each grouping together constraints with similar purposes.

a: TIMING CONSTRAINTS

In scheduling, it is crucial to establish timing boundaries for stream and task variables. A key variable is the stream duration, which needs to be set for each stream. The duration constraint of the stream, as shown in (8), varies from one link to another, depending on the link speed and the stream size. Other variables determined by the timing constraints include the end times of streams and tasks. As depicted in (9) and (10), the end times for both streams and tasks are calculated by adding an offset to the duration of each.

$$\begin{aligned}&\forall \sigma_i \in S, \forall \left(\upsilon_a, \upsilon_b\right) \in \mathcal{E} \cap \mathbb{R}^{\sigma}_{i} :\\ &\quad \varepsilon^{\sigma_i}_{(\upsilon_a,\upsilon_b)} = \left\lceil \frac{L^{\sigma}_{i}}{\Omega_{(\upsilon_a,\upsilon_b)}} \right\rceil \end{aligned} \tag{8}$$

$$\zeta^{\sigma}_{(\upsilon_a,\upsilon_b)} = \eta^{\sigma}_{(\upsilon_a,\upsilon_b)} + \varepsilon^{\sigma}_{(\upsilon_a,\upsilon_b)} \tag{9}$$

$$\begin{aligned}&\forall \tau_j \in T :\\ &\quad \zeta^{\tau}_{\upsilon_a} = \eta^{\tau}_{\upsilon_a} + \omega^{\tau}_{j} \end{aligned} \tag{10}$$

b: IN-LINKS TIMING CONSTRAINT

Ensuring that each routed stream follows the correct sequence of links is crucial for maintaining the intended transmission path. Equation (11) establishes a constraint on the timing of each stream across two sequentially connected links on its path. The constraint specifies that the offset of the stream on any given link must be equal to or greater than the end-time of that stream on the preceding link.

$$\begin{aligned}&\forall \sigma_i \in S, \forall \left(\upsilon_a, \upsilon_b\right), \left(\upsilon_b, \upsilon_c\right) \in \mathcal{E} \cap \mathbb{R}^{\sigma}_{i}, \upsilon_a = Z^{\sigma}_{\upsilon_b} :\\ &\quad \zeta^{\sigma}_{(\upsilon_a,\upsilon_b)} \leq \eta^{\sigma}_{(\upsilon_b,\upsilon_c)} \end{aligned} \tag{11}$$

c: PRIORITY SCHEDULING CONSTRAINTS

The presence of various stream types, each with different priorities, necessitates prioritizing scheduling times based on these priorities. Since bridge nodes are pivotal in stream transmission, having queued at the bridges' queues, it becomes essential to manage the opening and closing times of these queues. Equations (12) and (13) detail the constraints for opening and closing the queues' gates in accordance with their specific Queue-Gating Time (QGT), denoted as $\kappa^{q}_{\varphi}\boxed{\psi}$. These constraints affect the offset and end-time of each stream, ensuring transmission occurs during the open window of its respective queue at the bridge.

$$\begin{aligned}&\forall \sigma_i \in S, \forall br_n \in BR, \forall q^{br_n}_{\phi} \in Q^{br_n},\\ &\quad t^{br_n}_{\varphi} \in \Phi^{br_n}, \rho^{\sigma}_{i} == \rho^{q}_{\phi} :\\ &\eta^{\sigma_i}_{br_n} \geq \kappa^{q^{br_n}}_{\varphi}\boxed{1} \end{aligned} \tag{12}$$

$$\zeta^{\sigma_i}_{br_n} \leq \kappa^{q^{brn}}_{\varphi} \boxed{0} \tag{13}$$

d: NO OVERLAP CONSTRAINTS

Achieving a deterministic network behavior is a key objective, and avoiding overlaps in streams or tasks is essential for this. Equation (14) addresses this by preventing overlaps in streams that traverse a shared link in their routes. This constraint ensures that different streams use the link in distinct timeframes, thus avoiding simultaneous transmission. Similarly, potential overlaps in tasks executing on the same end-system node are addressed by (15). This constraint ensures that tasks are executed in separate timeframes, preventing any overlapping.

The concept of frame isolation in Time-Sensitive Networking (TSN) networks, first introduced in [45], plays a crucial role in this context. It specifically aims to prevent time frame overlaps for incoming streams to the same node, but from different links. These streams eventually share the same egress link from the node. Equation (16) outlines this constraint, and we provide a detailed explanation of it in our work [20].

$$\begin{aligned}
&\forall \sigma_1, \sigma_2 \in S, \sigma_1 \neq \sigma_2, \forall (\upsilon_a, \upsilon_b) \in \mathbb{R}^{\sigma}_1 \cap \mathbb{R}^{\sigma}_2, \\
&\quad \forall \alpha \in \left\{0, \cdots, \frac{lcm\left(\Gamma^{\sigma}_1, \Gamma^{\sigma}_2\right)}{\Gamma^{\sigma}_1}\right\}, \\
&\quad \forall \beta \in \left\{0, \cdots, \frac{lcm\left(\Gamma^{\sigma}_1, \Gamma^{\sigma}_2\right)}{\Gamma^{\sigma}_2}\right\} : \\
&\quad \left(\left(\alpha \times \Gamma^{\sigma}_1 + \zeta^{\sigma_1}_{(\upsilon_a,\upsilon_b)}\right) \leq \left(\beta \times \Gamma^{\sigma}_2 + \eta^{\sigma_2}_{(\upsilon_a,\upsilon_b)}\right)\right) \\
&\quad \vee \left(\left(\beta \times \Gamma^{\sigma}_2 + \zeta^{\sigma_2}_{(\upsilon_a,\upsilon_b)}\right) \leq \left(\alpha \times \Gamma^{\sigma}_1 + \eta^{\sigma_1}_{(\upsilon_a,\upsilon_b)}\right)\right)
\end{aligned} \tag{14}$$

$$\begin{aligned}
&\forall \tau_1, \tau_2 \in T, \tau_1 \neq \tau_2, \\
&\quad \forall \alpha \in \left\{0, \cdots, \frac{lcm\left(\Gamma^{\tau}_1, \Gamma^{\tau}_2\right)}{\Gamma^{\tau}_1}\right\}, \\
&\quad \forall \beta \in \left\{0, \cdots, \frac{lcm\left(\Gamma^{\tau}_1, \Gamma^{\tau}_2\right)}{\Gamma^{\tau}_2}\right\}, \\
&\quad \left(\left(\alpha \times \Gamma^{\tau}_1 + \zeta^{\tau_1}_{\upsilon_a}\right) \leq \left(\beta \times \Gamma^{\tau}_2 + \eta^{\tau_2}_{\upsilon_a}\right)\right) \\
&\quad \left(\left(\beta \times \Gamma^{\tau}_2 + \zeta^{\tau_2}_{\upsilon_a}\right) \leq \left(\alpha \times \Gamma^{\tau}_1 + \eta^{\tau_1}_{\upsilon_a}\right)\right)
\end{aligned} \tag{15}$$

$$\begin{aligned}
&\forall \sigma_1, \sigma_2 \in S, \sigma_1 \neq \sigma_2, \forall (\upsilon_b, \upsilon_c) \in \mathbb{R}^{\sigma}_1 \cap \mathbb{R}^{\sigma}_2, \\
&\quad \forall (\upsilon_{a1}, \upsilon_b) \in \mathbb{R}^{\sigma}_1, \forall (\upsilon_{a2}, \upsilon_b) \in \mathbb{R}^{\sigma}_2, \\
&\quad \upsilon_{a1} = Z^{\sigma_1}_{\upsilon_b}, \upsilon_{a2} = Z^{\sigma_2}_{\upsilon_b}, \\
&\quad \forall \alpha \in \left\{0, \cdots, \frac{lcm\left(\Gamma^{\sigma}_1, \Gamma^{\sigma}_2\right)}{\Gamma^{\sigma}_1}\right\}, \\
&\quad \forall \beta \in \left\{0, \cdots, \frac{lcm\left(\Gamma^{\sigma}_1, \Gamma^{\sigma}_2\right)}{\Gamma^{\sigma}_2}\right\} : \\
&\quad \left(\left(\alpha \times \Gamma^{\sigma}_2 + \eta^{\sigma_2}_{(\upsilon_b,\upsilon_c)}\right) \leq \left(\beta \times \Gamma^{\sigma}_1 + \eta^{\sigma_1}_{(\upsilon_{a1},\upsilon_b)}\right)\right) \\
&\quad \vee \left(\left(\beta \times \Gamma^{\sigma}_1 + \eta^{\sigma_1}_{(\upsilon_b,\upsilon_c)}\right) \leq \left(\alpha \times \Gamma^{\sigma}_2 + \eta^{\sigma_2}_{(\upsilon_{a2},\upsilon_b)}\right)\right)
\end{aligned} \tag{16}$$

e: TASK-STREAM COORDINATION CONSTRAINTS

It is crucial to define the relationship between streams and tasks. This interaction occurs at the end-system where tasks are executed. Streams are either sent from an end-system (talker) or received by one (listener). For outgoing streams from a talker, Equation (17) stipulates that the offset of the stream must be greater than or equal to the end-time of the talker's task. Conversely, for incoming streams to listeners, Equation (18) specifies that the end-time of the stream should be less than or equal to the offset of the task being executed on the listener.

These constraints apply specifically to streams and tasks that are part of the same application, meaning the stream is the communication link between the talker and listener tasks. This ensures that the data flow aligns with the task execution timings, maintaining the intended sequence and timing of operations within the system.

$$\begin{aligned}
&\forall \tau_j \in T, \forall \sigma_i \in S, \tau_j = \tau^{\sigma_i}_{talker}, \\
&\quad \forall (\upsilon_a, \upsilon_b) \in \mathcal{E} \cap \mathbb{R}^{\sigma}_i, \upsilon_a == es^{\tau}_j : \\
&\quad \zeta^{\tau}_{\upsilon_a} \leq \eta^{\sigma}_{(\upsilon_a,\upsilon_b)}
\end{aligned} \tag{17}$$

$$\begin{aligned}
&\forall \sigma_i \in S, \Pi^{\sigma}_i = 0, \forall \tau_j \in T, \tau_j \in T^{\sigma_i}_{listner}, \\
&\quad \forall (\upsilon_a, \upsilon_b) \in \mathcal{E} \cap \mathbb{R}^{\sigma}_i, \upsilon_b == es^{\tau}_j : \\
&\quad \zeta^{\sigma}_{(\upsilon_a,\upsilon_b)} \leq \eta^{\tau}
\end{aligned} \tag{18}$$

f: STREAM SCHEDULE RELIABILITY CONSTRAINTS

Earlier in the discussion on routing constraints, we introduced the stream path reliability constraint in (7), which mandates different paths for the original and redundant streams. Building upon this, in (19), we introduce an additional constraint designed to enforce a temporal shift between the original and redundant streams. This is achieved by establishing a rule: the offset time for the redundant stream σ_{iB} at link (υ_a, υ_b) should be greater than or equal to the end-time of the original stream σ_{iA} on a different link (υ_a, υ_c). Both of these links connect the talker end-system (υ_a) to two different bridges (υ_b) and (υ_c).

$$\begin{aligned}
&\forall \sigma_i \in S, \Pi^{\sigma}_i = 1, \\
&\quad \forall (\upsilon_a, \upsilon_b), (\upsilon_a, \upsilon_c) \in \mathcal{E} \cap \mathbb{R}^{\sigma}_i, \upsilon_a == es\langle \tau^{\sigma_i}_{talker} \rangle : \\
&\quad \eta^{\sigma_{iB}}_{(\upsilon_a,\upsilon_c)} \geq \zeta^{\sigma_{iA}}_{(\upsilon_a,\upsilon_b)}
\end{aligned} \tag{19}$$

$$\begin{aligned}
&\forall \sigma_i \in S, \Pi^{\sigma}_i = 1, \forall \tau_j \in T, \tau_j \in T^{\sigma_i}_{listner}, \\
&\quad \forall (\upsilon_a, \upsilon_c), (\upsilon_b, \upsilon_c) \in \mathcal{E} \cap \mathbb{R}^{\sigma}_i, \upsilon_c == es^{\tau}_j : \\
&\quad \begin{cases} \zeta^{\sigma_{iA}}_{(\upsilon_a,\upsilon_c)} \leq \eta^{\tau}, \text{if } \zeta^{\sigma_{iA}}_{(\upsilon_a,\upsilon_c)} \leq \zeta^{\sigma_{iB}}_{(\upsilon_b,\upsilon_c)} \\ \zeta^{\sigma_{iB}}_{(\upsilon_b,\upsilon_c)} \leq \eta^{\tau}, \text{if } \zeta^{\sigma_{iB}}_{(\upsilon_b,\upsilon_c)} < \zeta^{\sigma_{iA}}_{(\upsilon_a,\upsilon_c)} \end{cases}
\end{aligned} \tag{20}$$

With two different copies of the streams identified by $\Pi^{\sigma}_i = 1$ and a temporal shift between them, it's crucial to ensure that the listener task is initiated upon the receipt of either stream, while disregarding the other. This concept is introduced in Equation (20). The listener task's offset is timed to start after the end-time of the original stream σ_{iA} at the link connecting the listener node (υ_c) to the last bridge (υ_a) from which this stream originates. This approach is applied

if the original stream σ_{iA} arrives earlier than the redundant stream σ_{iB}, which originates from a different bridge (υ_b) and uses a different link to the listener node. The same principle applies in reverse: if the redundant stream σ_{iB} arrives first, the listener task will commence upon its receipt, ignoring the original stream σ_{iA}.

B. OBJECTIVE FUNCTIONS

In optimization-based problems, objective functions are vital for achieving the best results. Our framework model introduces two such functions. The first objective function aims to minimize the total length of the routing paths for the streams. By doing so, it ensures that streams travel the shortest possible path from their source to their destination, as detailed in (21). This approach reduces the distance each stream must cover, potentially leading to faster and more efficient communication.

$$min \sum_{\sigma_i \in S} \sum_{\forall \upsilon_a \in \mathcal{V} \ except \ \{es\langle \tau^{\sigma_i}_{listener} \rangle\}} (Z^{\sigma_i}_{\upsilon_a} ! = nil) \quad (21)$$

$$min \sum_{\delta_k \in \Delta} \left(\zeta^{\tau_{listner}} - \eta^{\tau_{talker}} \right),$$
$$where \ \tau_{listner}, \tau_{talker} \in T_k \quad (22)$$

The communication journey in the network starts with a task executed on the talker end-system, followed by a stream transmitted from this talker, passing through various bridges, and finally reaching the listener end-system where the corresponding listener task is executed. This entire process, spanning from the offset of the talker's task to the end-time of the listener's task, defines the latency of the communication. Minimizing this latency is crucial to ensure rapid communication. The second objective function, introduced in (22), focuses on minimizing the total latency of all transmissions. This is key to achieving a more optimal scheduling that aligns with the goals of Time-Sensitive Networking (TSN). By reducing the overall latency, the network becomes more efficient, capable of handling communication tasks in a timely and effective manner.

V. EXPERIMENTAL EVALUATION

The optimization framework presented will be evaluated in this section through the application of various problem instances. These instances will demonstrate its performance across multiple aspects.

A. PROBLEM INSTANCES AND EXPERIMENTAL SETUP

To evaluate the framework proposed in this paper, we will use eight distinct problem instances inspired by the network topologies of real-time applications. These instances vary in the number of bridges and end-systems, as detailed in Table 3. Half of the instances have an equal number of end-systems and bridges, while the other half have twice as many end-systems as bridges. This approach allows us to observe the impact of varying node type ratios on performance.

In the creation of network topologies for our instances, we devised an algorithm using the Python programming language and employed the NetworkX library [46], which assists in creating networks and network graphs. During the network construction of our instances, we ensured that no links connected two end-systems directly; the only permissible links were between bridges, and between a bridge and an end-system. We established a minimum of two random links between each bridge and the other bridges, and between an end-system and the bridges. This approach provided more potential paths from each node, aiding in the redundancy of the streams. The maximum number of links between each end-system and the bridges was set at four. Referring to links between a node and other nodes, it is understood that each link connects the node to one other node. Therefore, if an end-system has three links to bridges, it means this end-system is connected to three different bridges. The links are full-duplex, bi-directional, and have a transmission speed of 1 Gbps.

TABLE 3. Problem instance.

Instance Name	No. BRs	No. ESs	No. Tasks	No. TT Streams	No. BE Streams
Exemplary	4	4	7	2	1
06B06E	6	6	33	10	5
06B12E	6	12	42	12	6
12B12E	12	12	65	18	9
12B24E	12	24	98	28	14
18B18E	18	18	114	32	16
18B36E	18	36	130	36	18
24B24E	24	24	147	42	21
24B48E	24	48	174	48	24

The instance creation algorithm generates two types of streams: TT and BE streams, with a ratio of 2:1 for TT to BE streams. The number of TT streams for each problem instance scenario is deterministically assigned as an input parameter to the problem instance generation algorithm. For each application, one out of every four streams are randomly designated as multicast streams, while the remaining three are designated as unicast streams. Redundancy is applied stochastically to TT streams only; BE streams do not utilize redundancy. We have established three different redundancy scenarios based on the percentage of redundant TT streams. In the first scenario, 40% of the TT streams are redundant, while in the second scenario, this increases to 70%. The third scenario involves 100% of the TT streams being redundant. Each application contains only one type of stream, either TT or BE, with a minimum of four streams per application. The number of streams in each application influences the number of tasks, resulting in a range of 33 to 174 tasks for each instance. The number of tasks for each application is initialized stochastically. Each instantiation of the problem instance scenarios will result in a varying number of tasks. The Worst-Case Execution Time (WCET) of the tasks will be at maximum equal to three percent of their period, while the stream size, including frame overhead, will also be at maximum equal to the Maximum Transmission Unit

(MTU) of 1500*Bytes*. The period of the tasks, streams, and applications is set at 1000μs.

The bridges in our system are equipped with eight queues, each with a gate that opens and closes according to a predetermined GCL. The gates for the TT queues, which are the first two queues, open at the beginning of the Hyperperiod and close between 45% and 60% of the Hyperperiod. Conversely, the gates for the BE queues, which are the 7th and 8th queues, open between 60% and 70% of the Hyperperiod and close at the end of the Hyperperiod.

To set up our work, we utilized the CP-SAT Solver [47], a component of the Google OR-Tools suite. This solver employs Constraint Programming (CP) techniques to solve problems. We conducted the evaluation of the presented problem instances on a system powered by an Intel Core i7-11370H CPU, which runs at 3.30 GHz and is equipped with 8 CPUs and 8GB of memory.

B. LATENCY EVALUATION

Evaluating any TSN model hinges crucially on two key aspects: achieving optimal or feasible latency, and determining the worst-case latency (WCL) within the network. In this section, we will assess these parameters for all transmissions, encompassing both TT and BE applications. The Quality of Service (QoS) for real-time networks is primarily gauged by maintaining low latency and minimizing worst-case latency, ensuring efficient and reliable network performance.

For a comprehensive evaluation, we varied the redundancy percentage of the TT streams. Initially, a 40% redundancy was applied to the eight different problem instances of the TT streams. Subsequently, we increased this to 70% and finally to 100% redundancy. Table 4 presents the total latency of all network transmissions for the OHDSR+ model. The evaluation results indicate low latency across different redundancy percentages, with total latency being almost similar in some instances, even reaching a unified total latency. This was achieved by considering the first arrival among the redundant TT streams (whether original or redundant) and disregarding the other.

TABLE 4. The total latency for the problem instances run on the OHDSR+ model, when applying 40%, 70%, and 100% redundancy to the Time-Triggered (TT) streams.

Instance Name	Total Latency (μs) - OHDSR+		
	40% TT Redundancy	70% TT Redundancy	100% TT Redundancy
06B06E	772	772	773
06B12E	962	962	962
12B12E	1493	1496	1490
12B24E	2233	2237	2244
18B18E	2621	2633	2635
18B36E	2960	2963	2967
24B24E	3455	3451	3466
24B48E	3923	3922	3919

The Worst-case latency (WCL) for both TT and BE streams, considering a 40% redundancy of the TT streams, is shown in Table 5. These results demonstrate that the opening and closing of the gates, based on the priorities of the streams, result in a nearly identical WCL for both TT and BE streams.

TABLE 5. The worst-case latency for Time-Triggered (TT) streams and the worst-case latency for Best-Effort (BE) streams in the problem instances run on the OHDSR+ model, when applying 40% redundancy to the TT streams.

Instance Name	Worst-Case Latency (μs) - OHDSR+	
	TT Streams	BE Streams
06B06E	55	61
06B12E	61	60
12B12E	62	61
12B24E	62	58
18B18E	62	66
18B36E	65	62
24B24E	62	63
24B48E	61	63

In our research, we ran problem instances on our model that were also tested in previous related work. This work, which offers an open-source model, was presented by Reusch et al. [43]. We refer to this work as "REU-CP-R" for comparison purposes. Similar to our approach, REU-CP-R addressed reliability by tackling the joint scheduling and routing problem using a Constraint Programming (CP) based solver. Table 6 presents the total latency results for our problem instances when run on REU-CP-R across different redundancy percentages of Time-Triggered (TT) streams.

TABLE 6. The total latency for the problem instances run on the REU-CP-R model, when applying 40%, 70%, and 100% redundancy to the Time-Triggered (TT) streams.

Instance Name	Total Latency (μs) - REU-CP-R		
	40% TT Redundancy	70% TT Redundancy	100% TT Redundancy
06B06E	774	774	774
06B12E	966	962	972
12B12E	1494	1498	1502
12B24E	2237	2241	2251
18B18E	2638	2642	2652
18B36E	2971	2976	2984
24B24E	3473	3480	3482
24B48E	/	/	/

The results indicate that the REU-CP-R model achieved commendable total latency across all three scenarios. However, when compared with our OHDSR+ model, it becomes evident that OHDSR+ consistently achieves better minimization of total latency across all scenarios and instances. The extent of improvement in latency minimization with OHDSR+ varies, ranging from a 0.06% reduction in total latency for the 40% TT streams redundancy scenario in the 12B12E problem instance, to a 1.03% improvement for the 100% TT streams redundancy scenario in the 06B12E problem instance. Table 7 illustrates the percentage reduction

in total latency of the OHDSR+ model compared to the REU-CP-R model.

TABLE 7. The total latency minimization percentages of the OHDSR+ model compared to the REU-CP-R model for the problem instances, when applying 40%, 70%, and 100% redundancy to the Time-Triggered (TT) streams.

Instance Name	Total Latency Minimization Percentages		
	40% TT Redundancy	70% TT Redundancy	100% TT Redundancy
06B06E	0.26%	0.26%	0.13%
06B12E	0.41%	0.41%	1.03%
12B12E	0.06%	0.13%	0.80%
12B24E	0.18%	0.18%	0.31%
18B18E	0.64%	0.34%	0.64%
18B36E	0.37%	0.44%	0.57%
24B24E	0.52%	0.83%	0.46%
24B48E	/	/	/

It is important to note, as indicated in Tables 6 and 7, that for the 24B48E problem instance, the result is marked as "/". This denotes that the REU-CP-R model was unable to solve this instance for any of the redundancy scenarios. It returned an "UNKNOWN" value from the CP-SAT solver, indicating that no solution was found.

C. RESPONSE TIME EVALUATION

The time required for any optimization-based model to solve a specific problem and achieve an optimal or feasible solution is a critical factor. In our study, this time is referred to as the response time. We denote the total time taken to solve the problem as the 'total response time.' When discussing the time required for scheduling and routing, these terms will precede 'response time' to specify the duration needed for each model to address its respective problem. Similar to our latency evaluation approach, we assessed each problem instance with varying redundancy percentages of the TT streams: starting at 40%, then 70%, and finally evaluating the model's capability to handle 100% redundant TT streams.

Table 8 illustrates the total response times of our model for each problem instance under varying percentages of redundancy. The results indicate that the network's scale, along with the number of applications, tasks, and streams, significantly impacts the total response time. For instance, comparing the first two instances, we observe that while both have six bridges, the second instance includes an additional six end-systems, totaling twelve. Furthermore, this instance comprises nine more tasks, two additional TT streams, and one extra BE stream compared to the first. This leads to a substantial increase in response time for the second instance, almost tripling in the 40% TT redundancy scenario.

The redundancy percentage also plays a crucial role in determining the total response time. Taking the first instance as an example, the total response time increases by 17.15% and 44.96% for 70% and 100% redundancy of the TT streams, respectively, compared to the 40% scenario. Additionally,

TABLE 8. The total response time for the problem instances run on the OHDSR+ model, when applying 40%, 70%, and 100% redundancy to the Time-Triggered (TT) streams.

Instance Name	Total Response Time (μs) - OHDSR+		
	40% TT Redundancy	70% TT Redundancy	100% TT Redundancy
06B06E	694	813	1006
06B12E	1878	2026	2557
12B12E	5254	7125	7769
12B24E	13681	16386	19177
18B18E	94589	122018	78285
18B36E	170338	150950	176141
24B24E	160099	186924	232245
24B48E	555950	526974	954127

there is a 23.74% increase in response time when comparing the 100% redundancy scenario to the 70% scenario.

Although it is generally expected that larger-scale networks and instances with higher redundancy percentages will have longer total response times, this is not an absolute rule. The performance of a model solving a combinatorial optimization problem can be influenced by various factors, including the constraints, the problem's structure, and the complexity of the instances themselves. For example, instances created randomly could result in less complex network connections in larger-scale networks compared to smaller ones.

The impact of redundancy percentage on response time is another example. While a higher redundancy percentage typically leads to longer response times, this isn't always the case. As depicted in Fig 5, the instance with 18 bridges and 18 end-systems (18B18E) actually shows a shorter total response time for the 100% TT redundancy scenario compared to scenarios with lower percentages. Conversely, in the instance with 18 bridges and 36 end-systems (18B36E), the 70% TT redundancy scenario exhibits a shorter total response time than both the higher and lower percentage scenarios.

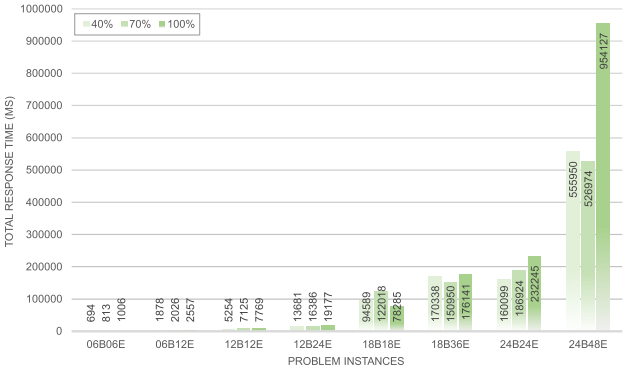


FIGURE 5. The total response time for the problem instances run on the OHDSR+ model, when applying 40%, 70%, and 100% redundancy to the Time-Triggered (TT) streams.

These variations can be attributed to multiple factors, such as the simplification of the problem, the search space,

and the specific structure of the instance under a particular redundancy scenario. When combined with the randomly created connections, these factors can sometimes make it easier for the model to solve a scenario with a higher redundancy percentage compared to others.

The response times of the scheduling and routing models are key in assessing the efficiency of each model independently. The OHDSR+ model, for example, demonstrates a notable response time in the scheduling model, reaching a maximum of 764.9 milliseconds for the 24B48E instance under a 100% TT redundancy scenario, as highlighted in Table 9. In contrast, as depicted in Table 10, the routing model exhibits a higher response time. This increased duration is attributed to the inherent complexity of routing problems, which involves complicated constraints and a large search space necessary to identify the optimal solution.

TABLE 9. The scheduling response time for the problem instances run on the OHDSR+ model, when applying 40%, 70%, and 100% redundancy to the Time-Triggered (TT) streams.

Instance Name	Scheduling Response Time (ms) - OHDSR+		
	40% TT Redundancy	70% TT Redundancy	100% TT Redundancy
06B06E	57.8	58.7	73.7
06B12E	68.0	83.1	92.9
12B12E	132.5	173.3	209.2
12B24E	194.7	255.4	294.4
18B18E	345.2	435.7	485.2
18B36E	291.5	414.1	514.0
24B24E	454.3	634.1	764.9
24B48E	455.4	627.8	688.8

TABLE 10. The routing response time for the problem instances run on the OHDSR+ model, when applying 40%, 70%, and 100% redundancy to the Time-Triggered (TT) streams.

Instance Name	Routing Response Time (ms) - OHDSR+		
	40% TT Redundancy	70% TT Redundancy	100% TT Redundancy
06B06E	577.0	666.1	825.6
06B12E	1611.1	1752.3	2168.0
12B12E	4724.0	6470.5	6945.4
12B24E	12085.0	14358.0	17005.0
18B18E	92572.8	119600.8	75341.1
18B36E	166367.7	145578.5	169897.2
24B24E	155769.3	181821.4	226220.5
24B48E	546980.6	516995.4	944054.5

Specifically, the longest routing response time recorded is 944,054.5 milliseconds, or approximately 15.7 minutes, for the 100% TT redundancy scenario in the 24B48E instance. This is in stark contrast to the scheduling response time for the same instance and redundancy scenario, which is only 688.8 milliseconds, less than a second. This significant difference underscores the complexity and challenges associated with the routing model compared to the scheduling model.

In our research, we compared our problem instances with the REU-CP-R model [43], a previous related work, for benchmarking purposes. The related work uses the term ''optimization time'' to refer to what we call ''response time.'' In our paper, we will use ''response time'' consistently to describe both works.

The OHDSR+ model demonstrated superior performance across all instances and in every TT redundancy scenario. An exception was the 24B48E problem instance, which the REU-CP-R model could not solve, as further explained in section V-D. The comparative analysis for the 40% TT redundancy scenario is presented in Fig 6. The results reveal that our model, OHDSR+, achieved shorter response times in comparison to the REU-CP-R model for all the instances evaluated. The greatest reduction in response time was 19.66% for the 12B12E instance, while the smallest reduction was 6.80% for the 12B24E instance, which corresponded to a 998ms faster processing in favor of OHDSR+.

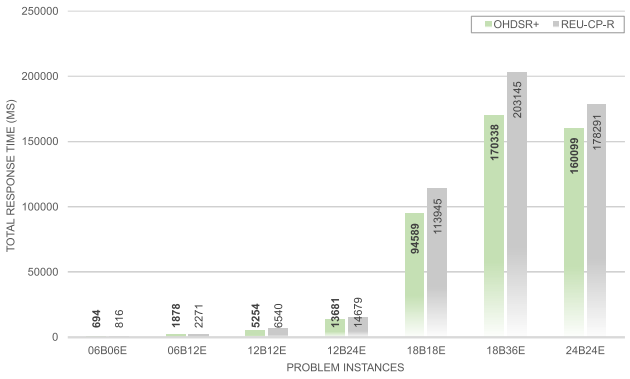


FIGURE 6. The total response time comparison between our proposed OHDSR+ model and the related work REU-CP-R [43] model when applying 40% redundancy to the Time-Triggered (TT) streams.

In Fig 7, a comparative analysis of the total response time results for the 70% TT redundancy scenario is showcased, highlighting the performance of our model, OHDSR+, against the REU-CP-R model. This comparison reveals that our model significantly outperforms the REU-CP-R model in most instances. Notably, in the 06B06E instance, OHDSR+ achieves a remarkable 34.70% reduction in total response time, marking the best minimization for this scenario. This demonstrates a clear advantage of our model in handling complex scheduling and routing optimization problems efficiently.

However, the comparison also shows varied performances across different instances. The least improvement using OHDSR+ is observed in the 18B18E instance, where the response time is reduced by only 1.60%. Despite being the minimal improvement, it's significant to note that this performance is still 1986 milliseconds faster compared to the REU-CP-R model, underscoring that even the lesser achievements of OHDSR+ are superior to the outcomes of the REU-CP-R model.

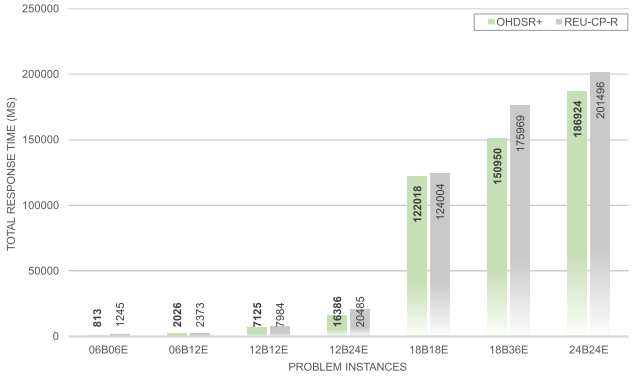


FIGURE 7. The total response time comparison between our proposed OHDSR+ model and the related work REU-CP-R [43] model when applying 70% redundancy to the Time-Triggered (TT) streams.

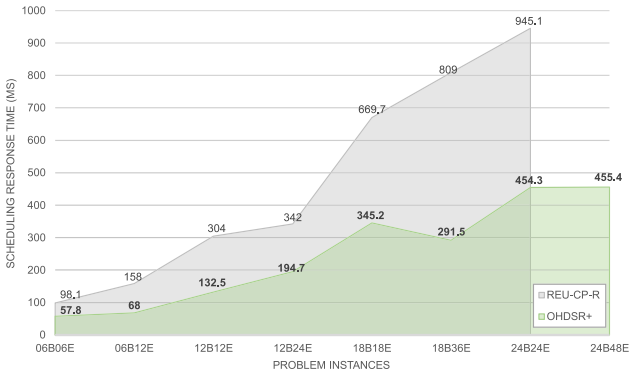


FIGURE 9. The scheduling response time comparison between our proposed OHDSR+ model and the related work REU-CP-R [43] model when applying 40% redundancy to the Time-Triggered (TT) streams.

The comparative analysis in Fig. 8 highlights the strengths of OHDSR+ across various scenarios and demonstrates its consistent efficiency over the REU-CP-R model, particularly in scenarios with 100% TT redundancy. Notably, OHDSR+ achieves an 18.92% reduction in the 24B24E instance. This detailed comparison further supports the evidence of OHDSR+'s effectiveness in optimizing response times across diverse network configurations.

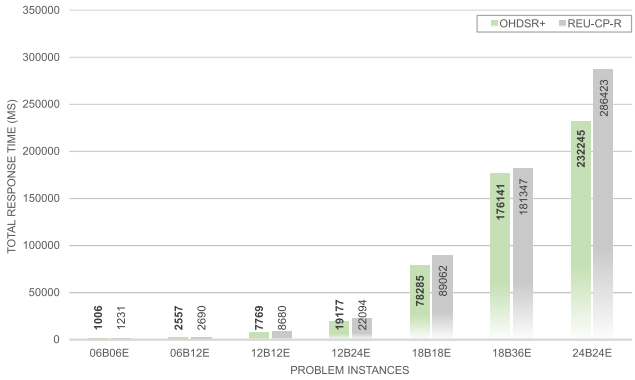


FIGURE 8. The total response time comparison between our proposed OHDSR+ model and the related work REU-CP-R [43] model when applying 100% redundancy to the Time-Triggered (TT) streams.

In addition to evaluating the total response time of our model and demonstrating its superior performance compared to related work, our study also focuses on scheduling, a key aspect of TSN standards. We conducted an evaluation of the scheduling model to compare its performance with the REU-CP-R model. Our findings indicate that our scheduling model, OHDSR+, outperforms the REU-CP-R model across all instances and in various redundancy scenarios.

This superior performance is evident in Fig 9, which presents a comparison of all instances with a 40% redundancy applied to the TT streams. The scheduling response time of our model shows a significantly better outcome, with reductions of up to 63.97%. This most notable reduction occurs in the 18B36E instance.

Furthermore, in the 06B06E instance, where the improvement is relatively lower, the scheduling response time is still 41.07% less than that of the REU-CP-R model. This reduction is particularly noteworthy when compared to the total response time, which shows a lesser improvement of only 6.80% for the same redundancy scenario. These results underscore the effectiveness of the OHDSR+ model, particularly in scheduling optimization, which is a crucial component of TSN standards implementation.

Figures 10 and 11 present the scheduling response times for the 70% and 100% TT stream redundancy scenarios, respectively. In these scenarios, a notably low scheduling response time is observed for the OHDSR+ model compared to the REU-CP-R model.

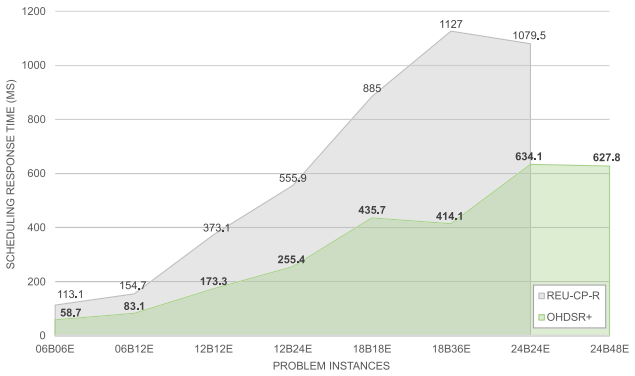


FIGURE 10. The scheduling response time comparison between our proposed OHDSR+ model and the related work REU-CP-R [43] model when applying 70% redundancy to the Time-Triggered (TT) streams.

For the 70% redundancy scenario, the difference in scheduling response time between OHDSR+ and REU-CP-R varies significantly, with OHDSR+ showing improvements ranging from 41.26% to 63.25%. This trend of superior performance is also evident in the 100% redundancy scenario, where OHDSR+ outperforms REU-CP-R by a margin of 45.42% to 63.73%.

A particularly notable point in these comparisons is the case of the 24B48E instance. This instance was only

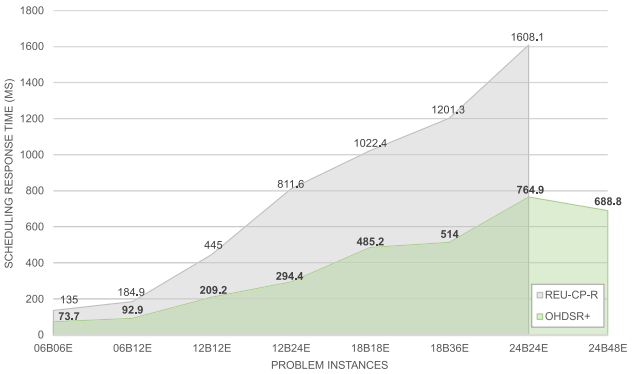


FIGURE 11. The scheduling response time comparison between our proposed OHDSR+ model and the related work REU-CP-R [43] model when applying 100% redundancy to the Time-Triggered (TT) streams.

successfully solved by our model, OHDSR+, as shown in the comparative analysis. The REU-CP-R model failed to find a solution for the routing model, and consequently, it could not address the scheduling model for this instance. This distinct outcome highlights the robustness and efficiency of the OHDSR+ model, especially in more complex scenarios where the challenge is significantly higher.

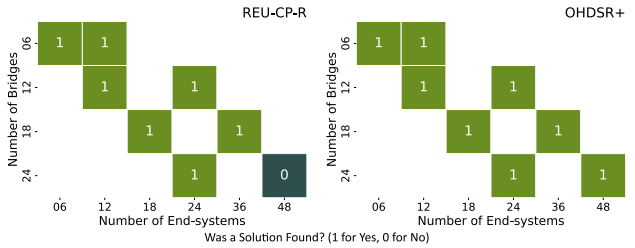


FIGURE 12. The scalability comparison between our proposed OHDSR+ model and the related work REU-CP-R [43] model.

D. SCALABILITY EVALUATION

To develop a feasible solution for our problem, it is crucial to evaluate scalability, which is one of the important aspects in TSN scheduling and routing optimization. Just as it is important to assess time, as we did in the two previous sections by evaluating the latencies of transmissions and the response time of the models, scalability is also an important evaluation metric. Scalability can be assessed by evaluating the model's ability to find feasible solutions for network topologies of varying scales or for different numbers of streams and tasks. The former is evaluated by testing the model with larger network topology sizes, increasing the number of devices, including bridges and end-systems, and observing the impact of larger scale networks on solution finding. The latter involves applying varying numbers of streams and tasks to a network topology of the same size and incrementally increasing these numbers to determine if the model can still find a feasible solution. As observed in our problem instances in Table 3, which feature various network topology sizes and numbers of streams and tasks, applying these instances to our model, OHDSR+, and the related work model, REU-CP-R, showed that our model was able to successfully find a solution in all instances with a 100% success rate. In contrast, the REU-CP-R model found solutions for all instances except the 24B48E instance and for all three different redundancy scenarios, achieving an 87.5% success rate. Fig 12 illustrates the scalability results, where a "1" indicates that the solver found a solution for the instance with the specified number of bridges and end-systems, a "0" indicates no solution was found for this instance, and an empty space indicates the absence of an instance with that specific number of devices.

VI. CONCLUSION

The scheduling and routing problem in Time-Sensitive Networking (TSN), with a focus on reliability, has been explored in this paper through the introduction of the OHDSR+ model. This model, based on constraint programming, addresses both Time-Triggered (TT) and Best-Effort (BE) traffic. Our work aims to schedule these different traffic types according to their priorities, controlled by a predefined Gate Control List (GCL) that manages the gate statuses assigned to the queues. Moreover, the reliability of TT traffic was considered by incorporating redundant paths for reliably transmitting TT traffic, along with a time shift to ensure a temporal difference between the original traffic and its redundant counterpart, thereby increasing traffic reliability and reducing the likelihood of data loss. The evaluation of our model demonstrates notable performance in terms of time and scalability. OHDSR+ achieves better results compared to similar approaches that consider reliability in scheduling and routing problems, evidenced by reduced latency times and improved response times. Another significant aspect of our model's performance is its scalability, which surpasses similar works by more effectively optimizing larger scale problem instances. Future research will focus on developing a more robust and resilient system by integrating security considerations directly into the scheduling and routing algorithms. This will involve exploring methodologies for securing redundant communication channels, including implementing encryption and authentication protocols specifically designed to prevent unauthorized access and data manipulation. Building upon the foundation established in this work, future research efforts aim to create a secure and resilient TSN ecosystem capable of effectively meeting the growing demands of real-time applications in an increasingly interconnected world.

REFERENCES

[1] *IEEE SA—The IEEE Standards Association*. Accessed: Jul. 8, 2023. [Online]. Available: https://standards.ieee.org/

[2] M. Vlk, Z. Hanzálek, K. Brejchová, S. Tang, S. Bhattacharjee, and S. Fu, "Enhancing schedulability and throughput of time-triggered traffic in IEEE 802.1Qbv time-sensitive networks," *IEEE Trans. Commun.*, vol. 68, no. 11, pp. 7023–7038, Nov. 2020, doi: 10.1109/TCOMM.2020.3014105.

[3] Y. Seol, D. Hyeon, J. Min, M. Kim, and J. Paek, "Timely survey of time-sensitive networking: Past and future directions," *IEEE Access*, vol. 9, pp. 142506–142527, 2021, doi: 10.1109/ACCESS.2021.3120769.

[4] *IEEE 802.1 AV Bridging Task Group*, IEEE Standard 802.1, Inst. Electr. Electron. Engineers, Jul. 2023. [Online]. Available: https://www.ieee802.org/1/pages/avbridges.html

[5] Institute of Electrical and Electronics Engineers. *IEEE 802.1 Working Group*. Accessed: Nov. 9, 2023. [Online]. Available: https://1.ieee802.org/

[6] *Time-Sensitive Networking Task Group*, IEEE Standards 802.1, Institute of Electrical and Electronics Engineers, Accessed: Nov. 9, 2023. [Online]. Available: https://www.ieee802.org/1/pages/tsn.html

[7] A. Nasrallah, A. S. Thyagaturu, Z. Alharbi, C. Wang, X. Shao, M. Reisslein, and H. Elbakoury, "Performance comparison of IEEE 802.1 TSN time aware shaper (TAS) and asynchronous traffic shaper (ATS)," *IEEE Access*, vol. 7, pp. 44165–44181, 2019, doi: 10.1109/ACCESS.2019.2908613.

[8] K. M. Shalghum, N. K. Noordin, A. Sali, and F. Hashim, "Worst-case latency analysis for AVB traffic under overlapping-based time-triggered windows in time-sensitive networks," *IEEE Access*, vol. 10, pp. 43187–43208, 2022, doi: 10.1109/ACCESS.2022.3168136.

[9] S. Vitturi, C. Zunino, and T. Sauter, "Industrial communication systems and their future challenges: Next-generation Ethernet, IIoT, and 5G," *Proc. IEEE*, vol. 107, no. 6, pp. 944–961, Jun. 2019, doi: 10.1109/JPROC.2019.2913443.

[10] K. M. Shalghum, N. K. Noordin, A. Sali, and F. Hashim, "Network calculus-based latency for time-triggered traffic under flexible window-overlapping scheduling (FWOS) in a time-sensitive network (TSN)," *Appl. Sci.*, vol. 11, no. 9, p. 3896, Apr. 2021, doi: 10.3390/app11093896.

[11] T. Stüber, L. Osswald, S. Lindner, and M. Menth, "A survey of scheduling algorithms for the time-aware shaper in time-sensitive networking (TSN)," *IEEE Access*, vol. 11, pp. 61192–61233, 2023, doi: 10.1109/ACCESS.2023.3286370.

[12] *IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks—Amendment 25: Enhancements for Scheduled Traffic*, IEEE Standard 802.1Q-2014/Cor 1-2015, 2016, pp. 1–57, doi: 10.1109/IEEESTD.2016.8613095.

[13] A. Nasrallah, A. S. Thyagaturu, Z. Alharbi, C. Wang, X. Shao, M. Reisslein, and H. ElBakoury, "Ultra-low latency (ULL) networks: The IEEE TSN and IETF DetNet standards and related 5G ULL research," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 88–145, 1st Quart., 2019, doi: 10.1109/COMST.2018.2869350.

[14] *IEEE Standard for Local and Metropolitan Area Networks—Virtual Bridged Local Area Networks Amendment 12: Forwarding and Queuing Enhancements for Time-Sensitive Streams*, IEEE Standard 802.1Qav-2009, 2010, pp. C1–72, doi: 10.1109/IEEESTD.2009.5375704.

[15] K. M. Shalghum, N. K. Noordin, A. Sali, and F. Hashim, "Critical offset optimizations for overlapping-based time-triggered windows in time-sensitive network," *IEEE Access*, vol. 9, pp. 130484–130501, 2021, doi: 10.1109/ACCESS.2021.3110585.

[16] *IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks—Amendment 26: Frame Preemption*, IEEE Standard 802.1Qbu-2016, 2016, pp. 1–52, doi: 10.1109/IEEESTD.2016.7553415.

[17] *IEEE Standard for Local and Metropolitan Area Networks-Frame Replication and Elimination for Reliability*, IEEE Standard 802.1CB-2017, 2017, pp. 1–102, doi: 10.1109/IEEESTD.2017.8091139.

[18] R. Hofmann, B. Nikolic, and R. Ernst, "Challenges and limitations of IEEE 802.1CB-2017," *IEEE Embedded Syst. Lett.*, vol. 12, no. 4, pp. 105–108, Dec. 2020, doi: 10.1109/LES.2019.2960744.

[19] *IEEE Standard for Local and Metropolitan Area Networks-Timing and Synchronization for Time-Sensitive Applications*, IEEE Standard 802.1AS-2011, 2020, pp. 1–421, doi: 10.1109/IEEESTD.2020.9121845.

[20] B. O. Akram, N. K. Noordin, F. Hashim, M. F. A. Rasid, M. I. Salman, and A. M. Abdulghani, "Joint scheduling and routing optimization for deterministic hybrid traffic in time-sensitive networks using constraint programming," *IEEE Access*, vol. 11, pp. 142764–142779, 2023, doi: 10.1109/ACCESS.2023.3343409.

[21] Y. Wang, J. Chen, W. Ning, H. Yu, S. Lin, Z. Wang, G. Pang, and C. Chen, "A time-sensitive network scheduling algorithm based on improved ant colony optimization," *Alexandria Eng. J.*, vol. 60, no. 1, pp. 107–114, Feb. 2021, doi: 10.1016/j.aej.2020.06.013.

[22] M. Pahlevan and R. Obermaisser, "Genetic algorithm for scheduling time-triggered traffic in time-sensitive networks," in *Proc. IEEE 23rd Int. Conf. Emerg. Technol. Factory Autom. (ETFA)*, vol. 1, Sep. 2018, pp. 337–344, doi: 10.1109/ETFA.2018.8502515.

[23] M. Pahlevan, N. Tabassam, and R. Obermaisser, "Heuristic list scheduler for time triggered traffic in time sensitive networks," *ACM SIGBED Rev.*, vol. 16, no. 1, pp. 15–20, Feb. 2019, doi: 10.1145/3314206.3314208.

[24] V. Gavrilut, L. Zhao, M. L. Raagaard, and P. Pop, "AVB-aware routing and scheduling of time-triggered traffic for TSN," *IEEE Access*, vol. 6, pp. 75229–75243, 2018, doi: 10.1109/ACCESS.2018.2883644.

[25] V. Gavrilut and P. Pop, "Scheduling in time sensitive networks (TSN) for mixed-criticality industrial applications," in *Proc. 14th IEEE Int. Workshop Factory Commun. Syst. (WFCS)*, Jun. 2018, pp. 1–4, doi: 10.1109/WFCS.2018.8402374.

[26] A. A. Atallah, G. B. Hamad, and O. A. Mohamed, "Fault-resilient topology planning and traffic configuration for IEEE 802.1Qbv TSN networks," in *Proc. IEEE 24th Int. Symp. On-Line Test. Robust Syst. Design (IOLTS)*, Jul. 2018, pp. 151–156, doi: 10.1109/IOLTS.2018.8474201.

[27] H. Li, H. Cheng, and L. Yang, "Reliable routing and scheduling in time-sensitive networks," in *Proc. 17th Int. Conf. Mobility, Sens. Netw. (MSN)*, Dec. 2021, pp. 806–811, doi: 10.1109/MSN53354.2021.00126.

[28] J. Falk, F. Dürr, and K. Rothermel, "Exploring practical limitations of joint routing and scheduling for TSN with ILP," in *Proc. IEEE 24th Int. Conf. Embedded Real-Time Comput. Syst. Appl. (RTCSA)*, Aug. 2018, pp. 136–146, doi: 10.1109/RTCSA.2018.00025.

[29] Q. Yu and M. Gu, "Adaptive group routing and scheduling in multicast time-sensitive networks," *IEEE Access*, vol. 8, pp. 37855–37865, 2020, doi: 10.1109/ACCESS.2020.2974580.

[30] E. Schweissguth, P. Danielis, D. Timmermann, H. Parzyjegla, and G. Mühl, "ILP-based joint routing and scheduling for time-triggered networks," in *Proc. 25th Int. Conf. Real-Time Netw. Syst.* New York, NY, USA: ACM, Oct. 2017, pp. 8–17, doi: 10.1145/3139258.3139289.

[31] E. Schweissguth, D. Timmermann, H. Parzyjegla, P. Danielis, and G. Mühl, "ILP-based routing and scheduling of multicast realtime traffic in time-sensitive networks," in *Proc. IEEE 26th Int. Conf. Embedded Real-Time Comput. Syst. Appl. (RTCSA)*, Aug. 2020, pp. 1–11, doi: 10.1109/RTCSA50079.2020.9203662.

[32] L. Xu, Q. Xu, J. Tu, J. Zhang, Y. Zhang, C. Chen, and X. Guan, "Learning-based scalable scheduling and routing co-design with stream similarity partitioning for time-sensitive networking," *IEEE Internet Things J.*, vol. 9, no. 15, pp. 13353–13363, Aug. 2022, doi: 10.1109/JIOT.2022.3143829.

[33] L. Yang, Y. Wei, F. R. Yu, and Z. Han, "Joint routing and scheduling optimization in time-sensitive networks using graph-convolutional-network-based deep reinforcement learning," *IEEE Internet Things J.*, vol. 9, no. 23, pp. 23981–23994, Dec. 2022, doi: 10.1109/JIOT.2022.3188826.

[34] H. Ali, U. U. Tariq, Y. Zheng, X. Zhai, and L. Liu, "Contention & energy-aware real-time task mapping on NoC based heterogeneous MPSoCs," *IEEE Access*, vol. 6, pp. 75110–75123, 2018, doi: 10.1109/ACCESS.2018.2882941.

[35] U. U. Tariq, H. Ali, L. Liu, J. Panneerselvam, and X. Zhai, "Energy-efficient static task scheduling on VFI-based NoC-HMPSoCs for intelligent edge devices in cyber-physical systems," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 6, pp. 1–22, Oct. 2019, doi: 10.1145/3336121.

[36] F. Pozo, G. Rodriguez-Navas, and H. Hansson, "Schedule reparability: Enhancing time-triggered network recovery upon link failures," in *Proc. IEEE 24th Int. Conf. Embedded Real-Time Comput. Syst. Appl. (RTCSA)*, Aug. 2018, pp. 147–156, doi: 10.1109/RTCSA.2018.00026.

[37] A. A. Atallah, G. B. Hamad, and O. A. Mohamed, "Routing and scheduling of time-triggered traffic in time-sensitive networks," *IEEE Trans. Ind. Informat.*, vol. 16, no. 7, pp. 4525–4534, Jul. 2020, doi: 10.1109/TII.2019.2950887.

[38] V. Balasubramanian, M. Aloqaily, and M. Reisslein, "Fed-TSN: Joint failure probability-based federated learning for fault-tolerant time-sensitive networks," *IEEE Trans. Netw. Service Manage.*, vol. 20, no. 2, pp. 1470–1486, Jun. 2023, doi: 10.1109/TNSM.2023.3273396.

[39] Z. Feng, C. Wu, Q. Deng, M. Han, and Y. Lin, "ReT-FTS: Re-transmission-based fault-tolerant scheduling in TSN," *J. Syst. Archit.*, vol. 142, Sep. 2023, Art. no. 102959, doi: 10.1016/j.sysarc.2023.102959.

[40] S. Hu, Y. Cai, S. Wang, and X. Han, "Enhanced FRER mechanism in time-sensitive networking for reliable edge computing," *Sensors*, vol. 24, no. 6, p. 1738, Mar. 2024, doi: 10.3390/s24061738.

[41] J. Min, W. Kim, J. Paek, and R. Govindan, "Effective routing and scheduling strategies for fault-tolerant time-sensitive networking," *IEEE Internet Things J.*, vol. 11, no. 6, pp. 11008–11020, Mar. 2024, doi: 10.1109/jiot.2023.3328626.

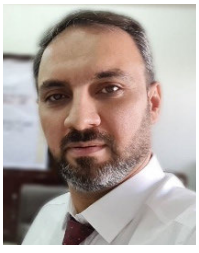
[42] M. Vlk, Z. Hanzálek, and S. Tang, "Constraint programming approaches to joint routing and scheduling in time-sensitive networks," *Comput. Ind. Eng.*, vol. 157, Jul. 2021, Art. no. 107317, doi: 10.1016/j.cie.2021.107317.
[43] N. Reusch, S. S. Craciunas, and P. Pop, "Dependability-aware routing and scheduling for time-sensitive networking," *IET Cyber-Phys. Syst., Theory Appl.*, vol. 7, no. 3, pp. 124–146, Sep. 2022, doi: 10.1049/cps2.12030.
[44] Q. D. Pham and Y. Deville, "Solving the quorumcast routing problem by constraint programming," *Constraints*, vol. 17, no. 4, pp. 409–431, Oct. 2012, doi: 10.1007/s10601-012-9125-z.
[45] S. S. Craciunas, R. S. Oliver, M. Chmelík, and W. Steiner, "Scheduling real-time communication in IEEE 802.1Qbv time sensitive networks," in *Proc. 24th Int. Conf. Real-Time Netw. Syst.*, Oct. 2016, pp. 183–192, doi: 10.1145/2997465.2997470.
[46] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using NetworkX," in *Proc. 7th Python Sci. Conf.*, G. Varoquaux, T. Vaught, J. Millman, Eds., Pasadena, CA USA, 2008, pp. 11–15.
[47] Google for Developers. *CP-SAT Solver | OR-Tools*. Accessed: Jul. 20, 2023. [Online]. Available: https://developers.google.com/optimization/cp/cp_solver

BILAL OMAR AKRAM received the B.Sc. degree in computer and information engineering from the University of Mosul, Nineveh, Iraq, and the M.Sc. degree in communication engineering from Universiti Putra Malaysia, where he is currently pursuing the Ph.D. degree. His research interests include wireless communications and network systems, time-sensitive networking (TSN), constraint programming (CP) techniques, artificial intelligence (AI), machine learning, deep learning, and the optimization, performance analysis, and evaluation of real-time networks.

NOR KAMARIAH NOORDIN (Senior Member, IEEE) received the B.Sc. degree in electrical engineering from The University of Alabama, Tuscaloosa, AL, USA, in 1987, the M.Eng. degree from Universiti Teknologi Malaysia, and the Ph.D. degree from Universiti Putra Malaysia. She is currently a Professor with the Department of Computer and Communication System Engineering, Universiti Putra Malaysia. She has published more than 300 journals, book chapters, and conference papers. She has led many research projects. Her research interests include wireless communications and network systems.

FAZIRULHISYAM HASHIM (Member, IEEE) received the B.Eng. degree in computer and communication systems from Universiti Putra Malaysia, the M.Sc. degree from Universiti Sains Malaysia, and the Ph.D. degree in wireless communication and networks engineering from The University of Sydney, Australia. He is currently an Associate Professor with the Department of Computer and Communication Systems Engineering, Universiti Putra Malaysia. He has published more than 150 journals, book chapters, and conference papers. His research interests include heterogeneous and future wireless communication systems and network security.

MOHD FADLEE A. RASID received the B.Sc. degree in electrical engineering from Purdue University, USA, and the Ph.D. degree in electronic and electrical engineering (mobile communications) from Loughborough University, U.K. He is currently with the Faculty of Engineering, Universiti Putra Malaysia (UPM). He directs research activities within the Wireless Sensors Group and his work on wireless medical sensors has gained importance in healthcare applications involving mobile telemedicine and has had worldwide publicity, including BBC News. He was a Research Consultant of the British Council UKIERI Project on wireless medical sensors project. He was also part of the French Government STIC Asia Project on ICT-ADI: Toward a Human-Friendly Assistive Environment for Aging, Disability, and Independence. He had led a few projects on WSN from the Ministry of Communications and Multimedia Malaysia and the Economic Planning Unit (EPU) under the Prime Minister's Department, particularly for agriculture and environmental applications. He was involved with a project under Qatar National Research Fund by Qatar Foundations on Ubiquitous Healthcare and recently with Korean Development Institute (KDI) on Smart City. Professionally, he has been actively involved as the Lead Auditor for Integrated Rating for University and University College Excellence (SETARA) and Malaysian Quality Evaluation System for private colleges (MyQUEST).



MUSTAFA ISMAEL SALMAN received the B.Sc. degree in electronic and communication engineering and the M.Sc. degree in modern communications engineering from the University of Al-Nahrain, Baghdad, Iraq, in 1997 and 2000, respectively, and the Ph.D. degree in computer and communications engineering—wireless communications from Universiti Putra Malaysia, in 2015. He is currently an Assistant Professor with the Department of Computer Engineering, College of Engineering, University of Baghdad. His research interests include wireless communications, network virtualization, cloud computing, and software-defined networks.

ABDULRAHMAN M. ABDULGHANI (Member, IEEE) received the B.Sc. degree in computer science from Al-Rafidain University College, Baghdad, Iraq, in 2009, and the M.Sc. degree in distributed computing and networks from Universiti Putra Malaysia, in 2021, where he is currently pursuing the Ph.D. degree, with a focus on software-defined wide area networks (SD-WAN). His research interests include wireless communication, parallel and distributed computing, software-defined networking (SDN), SD-WAN, autonomous systems, security in computing, machine learning, and deep learning.