IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 25 October 2023, accepted 7 December 2023, date of publication 15 December 2023,
date of current version 21 December 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3343409

RESEARCH ARTICLE

Joint Scheduling and Routing Optimization for Deterministic Hybrid Traffic in Time-Sensitive Networks Using Constraint Programming

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], (Student 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 Computer Science, Faculty of Computer Science and Information Technology, University Putra Malaysia (UPM), Serdang, Selangor 43400, Malaysia

Corresponding authors: Bilal Omar Akram (bilal9omar2@gmail.com) and Nor Kamariah Noordin (nknordin@upm.edu.my)

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

ABSTRACT Real-time communications characterized by low-latency, deterministic, and reliable behavior are crucial for the advancement of emerging technologies. Consequently, Time-Sensitive Networking (TSN) has been developed to address the distinct demands of sectors such as automation and autonomous vehicles applications. This is currently achieved through various methods that emphasize the scheduling of critical data traffic. However, many of these methods determine routes independently, potentially impacting the schedulability of transmissions. Additionally, there is a noticeable lack of emphasis on the scheduling and routing of low-priority transmissions within TSN. In this paper, we introduce the Optimized Hybrid Deterministic Scheduling and Routing (OHDSR) approach. This method takes into account the priority of communications to jointly optimize the scheduling and routing of Time-Triggered (TT) communications, while also catering to low-priority Best-Effort (BE) communications. Extensive experimental evaluations show the high efficiency of our proposed method. It ensures not only the prompt delivery of TT communications but also the delivery of BE communications within suitable time frames, with a maximum difference of 14.29% from TT communications, meeting their respective deadlines. Moreover, the evaluation demonstrates the high scalability of the proposed approach, providing improved response times compared to the latest work for both routing and scheduling.

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

I. INTRODUCTION

The recent expansion in networking and its integration with new industries has led to the development of complex networks that are not adequately equipped for real-time communications. Network protocols must excel in scalability, cost, compatibility, safety, and real-time application handling. Although Ethernet is a well-established network protocol, it is not well-suited for real-time applications [1]. In recent years, these features have become critical for modern networks and applications, including those used in automation industries and autonomous vehicles. To address these challenges, various Ethernet-based protocols have been proposed, including Time-Triggered (TT) Ethernet and Audio/Video Bridging (AVB). However, none of these later protocols fully meet the requirements of new real-time applications [2].

The associate editor coordinating the review of this manuscript and approving it for publication was Bijoy Chand Chatterjee.

A new technology, Time-Sensitive Networking (TSN), was introduced, building on TT-Ethernet and AVB to offer enhanced real-time features for Ethernet networks [3].

Building on various extensions related to TT-Ethernet and AVB, the Institute of Electrical and Electronics Engineers Standards Association (IEEE SA) [4] initiated the IEEE 802.1 Time-Sensitive Networking (TSN) Task Group (TG). This group has established a set of standards that enhance Ethernet capabilities. These standards introduce new features that bolster the robustness and determinism of the networks. Additionally, they enable real-time synchronization and allow for effective management of different traffic types [5]. The TSN framework is designed to handle hybrid traffic encompassing three types, each with distinct priorities for mixed-criticality applications [6]. The highest priority class, which requires bounded latency and zero-jitter, is the TT traffic. This is articulated in IEEE 802.1Qbv [7]. The mechanism employed to configure the TT traffic in the Gate Control List (GCL) schedule is known as Time-Aware Shaping (TAS). While TAS manages the highest-priority class, the Credit-Based Shaping (CBS) mechanism configures the subsequent highest traffic class: the AVB traffic. CBS was introduced in IEEE 802.1Qav [8]. The final TSN traffic class is Best-Effort (BE). Since there is no need for a timing guarantee for this traffic type, it is regarded as the lowest priority class.

Scheduling transmissions with real-time guarantees is imperative to achieve minimal latency for different types of traffic. It's crucial to design appropriate Gate Control Lists (GCLs) for each Bridge (BR) to ensure the transmission of TT traffic, while also taking into account low-priority traffic, such as BE traffic. Thus, the contributions of this paper can be summarized as follows:

- We introduce a model called optimized hybrid deterministic scheduling and routing (OHDSR). This model addresses the routing and scheduling of time-triggered traffic while also considering best-effort traffic based on their respective priorities;
- We are unveiling an innovative bridge model that uses a Queue-Gating Time (QGT) approach. This method queues various traffic types and determines the specific times to open and closed the gates;
- As part of the OHDSR model, we incorporate a priority stream constraint designed to regulate the gating mechanism at the egress port of every bridge.

The structure of this paper is organized as follows: Section II introduces the TSN background and discusses related work. Section III defines the system models. In Section IV, we present our optimization framework, detailing the constraints and objective functions. The results of our evaluations are discussed in Section V. The paper wraps up with conclusions in Section VI.

II. BACKGROUND AND RELATED WORKS

Time-Sensitive Networking (TSN) is an emerging technology that has garnered considerable attention in the fields of computer networks and communications research over the past decade. While not as widely recognized as Ethernet technology, a clear overview of TSN is essential. In this section, we will delve into the background of TSN and survey relevant studies in the area.

A. TIME-SENSITIVE NETWORKING (TSN)

For many years, Ethernet has dominated as the primary technology in networking infrastructure. Its straightforward installation and robust performance have cemented it as one of IEEE SA's most successful standardizations. However, for a long time, Ethernet lacked support for real-time communication—a critical requisite in sectors such as autonomous vehicles, avionics, and industrial automation. This gap led to the emergence of various protocols tailored for specific applications: CAN and FlexRay for autonomous vehicles; Avionics Full-Duplex Switched Ethernet (AFDX) for avionics; and EtherCAT, PROFINET, and Sercos III for industrial contexts. While these protocols are adept at their designated functions, they often encounter compatibility issues when integrated with Ethernet-centric networks, and their adaptability across different industries is limited [9]. Recognizing these challenges, the IEEE Time-Sensitive Networking (TSN) task group has been proactively standardizing a technology equipped for real-time communication in the recent past.

TSN enhances Ethernet by enabling deterministic communication, ensuring that high-priority traffic is delivered within a defined time frame. The journey towards TSN can be traced back to the IEEE 802.1 Audio Video Bridging (AVB) standards, which introduced several features later adopted by TSN. Initially, the Audio Video Bridging Task Group was part of the IEEE 802.1 working group. However, it was renamed to ''Time-Sensitive Networking Task Group'' in November 2012 [10].

TSN incorporates various features, some of which originated from AVB, such as traffic shaping and scheduling, time synchronization, network management, and stream reliability. Maintaining the Quality of Service (QoS) for time-sensitive applications is paramount given their synchronization demands and time constraints. As such, TSN ensures QoS by integrating standardized features designed for specific objectives [11]. Numerous mechanisms have been introduced under different TSN standards: IEEE 802.1Qav [8] introduced the Credit-based Shaper (CBS), while IEEE 802.1Qbv [7] detailed the Time-Aware Shaper (TAS) and the Gate Control List (GCL). The significance of timing for real-time applications and synchronization across various network components was addressed in the IEEE 802.1AS standard [12] Another notable mechanism is frame preemption, which momentarily interrupts the transmission of low-priority traffic in favor of high-priority traffic. This was proposed in IEEE802.1Qbu [13]. Though not the final standard devised by the TSN task group, IEEE802.1CB [14], which introduces the Frame Replication and Elimination for Reliability (FRER) mechanism, is worth mentioning.

The Time-Aware Shaper (TAS) schedules time-critical streams within Time-Triggered (TT) windows, often referred to as protected traffic windows or time-aware traffic windows [15]. TAS traffic classes are predetermined based on the priority code point (PCP) values of the Virtual Local Area Network (VLAN) ID (VID) tag in 802.1Q frames [5]. In TSN, hybrid communication is allocated for three distinct traffic classes: Time-Triggered (TT) traffic, Audio Video Bridging (AVB) traffic, and Best Effort (BE) traffic [16]. The scheduling of these traffic types is governed by the Gate Control List (GCL), a list containing entries that dictate whether each port is open or closed and which type of traffic can be sent out through open-status ports [17]. Both end-systems and bridges harnessing the capabilities of TSN allocate a series of queues to each egress port. The number of these queues varies, contingent on the specific features of each device.

To ensure all devices function harmoniously, they must be synchronized according to a global time standard, achieved using a reliable clock synchronization technique such as IEEE 802.1ASrev. Subsequent to this synchronization, gates are situated at the end of each queue, with their operation governed by the Gate Control List (GCL). The Time-Aware Shaper (TAS) utilizes the GCL to guarantee unobstructed access to the outgoing port for each TT stream. By adeptly integrating clock synchronization and TAS, traditional Ethernet networks are primed to support real-time applications requiring minimal latency and jitter [18]. Consequently, crafting an accurate Gate Control List (GCL) for each specific scenario is vital to fully tap into the potential of TSN within a network.

B. RELATED WORKS

The Time-Sensitive Networking (TSN) standards, such as IEEE 802.1Qbv, define scheduling mechanisms. However, these standards do not specify any particular scheduling approaches. In this section, we will explore research studies related to TSN scheduling, with a special focus on those that address joint routing. According to a survey by [3], two main categories of algorithms have been introduced for TSN scheduling. The first category comprises Exact approaches, such as Integer Linear Programming (ILP), Satisfiability Modulo Theories (SMT), and Constraint Programming (CP). These methods can take considerable time but aim to find an optimal schedule with a specific objective. The second category, the Heuristic approach, provides a reasonably good solution more quickly than the Exact methods. Examples of these methods are the Greedy Randomized Adaptive Search Procedure (GRASP), Tabu Search, Simulated Annealing (SA), List Scheduling (LS), and Genetic Algorithms (GA).

Many researchers have investigated the joint routing and scheduling problem using various approaches. The ILP method was introduced in [19] to address this problem. In their evaluation, the researchers considered the effects of graph size, the number of streams, topology, and transmission frequency. Their optimization method's computation time was significantly influenced by the number of streams, whereas the network topology size had a lesser impact. Meanwhile, in [20], the authors reduced computational complexity and enhanced scalability in large-scale networks using an SMT-based approach.

Some researchers have employed heuristic algorithms in their joint routing and scheduling investigations. For instance, Wang et al. [21] utilized Ant-Colony Optimization (ACO), whereas another study [22] adopted Genetic Algorithms (GA).

The aspect of reliability was considered in [23], where the topology synthesis problem was formulated as an iterative path selection issue to minimize network costs. In another study [24] a CP-based approach was presented, introducing two Constraint Programming (CP) models: one featuring simple disjunctions, and the other incorporating optional interval variables which denote frame waiting times in queues. The methodology is as follows: first, a routing for the given streams is computed. A schedule is then determined based on this routing. If a schedule isn't identified, constraints are added to the routing model to prevent the previous routing solution. This process is repeated until a schedule is finalized. However, it is possible that all potential routings for some streams might result in scheduling conflicts. Considering both reliability and security, a CP model, along with a combined SA and LS model, was introduced in [25]. This research accounted for task scheduling and multicast stream scheduling.

Other studies, like those by Schweissguth et al. [26], [27] and Yu et al. [28], have based their multicast stream scheduling on ILP. Similarly, Pahlevan et al. [18], [29] employed heuristic strategies in their scheduling, ensuring the multicast nature of the streams was addressed.

While most previous studies have concentrated on the scheduling and routing of TT traffic, some researchers have also considered other types of traffic, such as AVB and BE. Gavrilut et al. [30], [31], [32] have developed various heuristic algorithms to schedule TT streams, while simultaneously accounting for AVB traffic. Moreover, a machine learning approach based on deep reinforcement learning was introduced by Yang et al. [33]. This approach takes into account TT traffic as well as both AVB and BE traffic. In our study, we are exploring a constraint programming (CP)-based approach to joint routing and scheduling. This approach considers factors like multicast capabilities and BE traffic.

III. SYSTEM MODEL

In this section, we will introduce our network, application, and bridge models. We will also explain the problem in detail. The notations used are provided in Table 1.

A. NETWORK MODEL

Our network model can be represented as a directed graph $\mathcal{G} =< \mathcal{V}, \mathcal{E} >$, where $\mathcal{V}$ denotes nodes and $\mathcal{E}$ signifies edges. The nodes $v_a \in \mathcal{V}$ can be Bridges (BR), also known

TABLE 1. Notations.

Symbol	Description
HP	Hyperperiod
$\mathcal{G} =< \mathcal{V}, \mathcal{E} >$	Network graph
$\upsilon_a \in \mathcal{V}$	Vertex (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)
$\eta_{(\upsilon_a,\upsilon_b)}^\sigma$	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

as network switches, or end-systems (ES). A connection between any two nodes, (υ_a, υ_b), establishes a full-duplex and bi-directional edge or link. Within this framework, the sender end-system is termed as the ''talker,'' and the receiver end-system is labeled as the ''listener.''

It is assumed that our entire network, including every node, is synchronized. Every node is equipped with complete TSN capabilities and supports the 802.1Qbv standard, leveraging the Time-Aware Shaper (TAS) to manage traffic. Each end-system node, υ_a, has a task scheduler governed by a periodic table.

B. APPLICATION MODEL

Our application model, $\delta_k \in \Delta$, is represented as a directed acyclic graph. It consists of a set of nodes that represent tasks τ_j. Concurrently, a set of edges, $\mathcal{E}$, denotes the data communication between these tasks. Each application is characterized by its own period Γ_k^δ. The Hyperperiod (HP) is derived by determining the least common multiple (lcm) of all these periods, given by: $HP = \text{lcm}(\{\Gamma_k^\delta | \delta_k \in \Delta\})$.

An application task $\tau_j \in T$ can be defined by the tuple (es_j^τ, ω_j^τ, Γ_j^τ), where es_j^τ denotes the execution end-system, ω_j^τ represents the worst-case execution time (WCET), and Γ_j^τ indicates its period. The set T contains all such tasks. The initial end-system sending the stream is termed a 'talker', while the receiver end-system assumes the role of a 'listener'. Upon completing source task execution, the talker (or sender end-system) produces outgoing data streams, while the listener (or receiver end-system) must receive all incoming data streams before it can begin destination task execution. When tasks operate on the same end-system, their communication dependencies are typically managed through message queues or shared memory. However, for tasks on different end-systems, their communication requirements create dependencies represented by streams. In the TSN context, a stream denotes a communication need between a talker and one or more listeners, often referred to as unicast or multicast.

We begin by examining multicast streams. A stream, denoted as $\sigma_i \in S$ originates from a source task labeled $\tau_{talker}^{\sigma_i}$ and is directed to multiple destination tasks, $T_{listner}^{\sigma_i}$. The set S encompasses all such streams. Each stream necessitates a designated path (or route) to ensure it reaches its intended destination within its period Γ_i^σ. The Maximum Transmission Unit (MTU) represents the upper limit of the size of each stream, L_i^σ, and is predefined for the network.

C. BRIDGE MODEL

In this section, we describe the architecture of the bridges used within our network. A consistent design is employed across all bridges. Each bridge, denoted as br_n and part of the set BR (which encompasses all bridges in the network), possesses eight distinct queues, represented as $q_\phi^{br_n}$ and belonging to the set Q^{br_n}. To streamline our representation and given that all bridges share the same configuration, we will exclude the br_n notation from the subsequent notations mentioned. Each queue is assigned a specific type, denoted by ρ_ϕ^q, indicating the nature of traffic it manages. When the type of a stream, identified as ρ_i^σ, matches the type associated with one or more queues, the stream is allocated to those queues based on current queue storage conditions.

For example, in our configuration, we allocate queues q_0 and q_1 specifically to manage TT traffic. Suppose q_0 currently handles three TT streams while q_1 manages two. The next incoming TT stream will be directed to q_1. If, however, both queues are serving an equal number of streams, the subsequent TT stream will be assigned to the queue with the earlier order — in this case, q_0.

Each queue is paired with a specific gate, denoted as $g_q \in G_q$, responsible for regulating the outgoing data flow from the queue. These gates can assume one of two states: open or closed, determined by predefined Gate Control Lists (GCLs). Initially, all gates are set to the closed state. Every GCL has an associated gating time, represented as $t_\varphi \in \Phi$, during which the gate state for each queue, symbolized as ψ_g, alters according to the respective GCL entry. This GCL entry is represented by an eight-digit binary number: ones (1s) indicate an open gate, and zeros (0s) a closed one. The binary

number is read from right to left: the far-right digit corresponds to the first queue q_0, and it progresses sequentially to the eighth digit that represents the last queue, q_7.

Queues are categorized as either Time-Triggered (TT) or Best-Effort (BE). Each type is tailored to handle specific traffic streams; TT queues manage TT streams, while BE queues handle BE streams. In the current setup, q_0 and q_1 are allocated for TT traffic, while q_6 and q_7 cater to BE traffic. The intermediate four queues from q_2 to q_5 remain unutilized and are tagged as " N/U " (Not Used).

This queue configuration remains unchanged across different scenarios. Consequently, there will be four gate state transitions. Initially, the TT queues open at gating time one, $t_1 \in \Phi$ with a GCL entry of "00000011". All gates then closed at gating time two, t_2, evidenced by a GCL entry of "00000000". The BE traffic gates activate at gating time three, t_3, with a GCL entry of "11000000". All gates eventually closed at gating time four, t_4, as indicated by a GCL entry of "00000000".

In some cases, BE traffic gates may open concurrently with the closure of the TT traffic gates. In such a scenario, the first gating time activates the TT traffic gates, the subsequent interval simultaneously deactivates TT traffic gates and activates the BE traffic gates, and the final interval deactivates all gates. The gating times, as defined by our GCLs, are influenced by the network applications' Hyperperiod to enhance the likelihood of optimal network scheduling. Fig. 1 visually presents this bridge design.

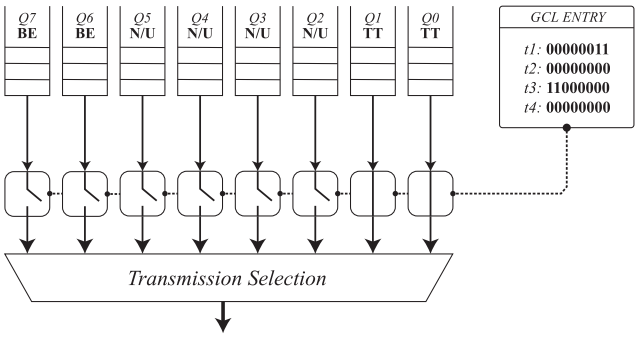


FIGURE 1. A simplified TSN bridge features queues and gates, Time-Triggered (TT) traffic is allocated to Queue 0 and 1, while Best Effort (BE) traffic is in Queue 6 and 7. Queues 2-5 are unused (N/U).

We are modeling the Queue-Gating Time (QGT), represented as $\kappa_{\varphi}^{q}\boxed{\psi}$. This model's derivation stems from the gating time $t_{\varphi} \in \Phi$ associated with each bridge.

In this setup, the GCL entry "00000011" signifies that at $t_1 = 0\mu s$, the gates of q_0 and q_1 will be open (indicated by 1). In contrast, the gates from q_2 to q_7 will be closed (indicated by 0). As a specific example, when $t_1 = 0\mu s$, q_0 has an open status, expressed as $\psi_0 = 1$, denoting $\kappa_1^0\boxed{1} = 0\mu s$. Similarly, q_5 remains closed, expressed as $\psi_5 = 0$ at $t_1 = 0\mu s$, resulting in $\kappa_1^5\boxed{0} = 0\mu s$.

The QGT values, represented as $\kappa_{\varphi}^{q}\boxed{\psi}$, are tabulated in Table 2. This table is structured as a matrix: rows illustrate the queues Q, and columns correspond to Gating Times Φ. The matrix denotes that at a specific gating time, termed t_{φ}, the value $\kappa_{\varphi}^{q}\boxed{\psi}$ matches that particular gating time. Thus, $\kappa_{\varphi}^{q}\boxed{\psi}$ equals t_{φ}. and relates to the queue q_{ϕ} with gate status ψ_g.

TABLE 2. A Queue-Gating Time (QGT) determines when each queue is open (indicated by a status of "1") or closed (indicated by a status of "0") at a particular gating time.

$\kappa_{\varphi}^{q}\boxed{\psi}$	t_1	t_2	t_3	t_4
q_0	$\kappa_1^0\boxed{1}$	$\kappa_2^0\boxed{0}$	$\kappa_3^0\boxed{0}$	$\kappa_4^0\boxed{0}$
q_1	$\kappa_1^1\boxed{1}$	$\kappa_2^1\boxed{0}$	$\kappa_3^1\boxed{0}$	$\kappa_4^1\boxed{0}$
q_2	$\kappa_1^2\boxed{0}$	$\kappa_2^2\boxed{0}$	$\kappa_3^2\boxed{0}$	$\kappa_4^2\boxed{0}$
q_3	$\kappa_1^3\boxed{0}$	$\kappa_2^3\boxed{0}$	$\kappa_3^3\boxed{0}$	$\kappa_4^3\boxed{0}$
q_4	$\kappa_1^4\boxed{0}$	$\kappa_2^4\boxed{0}$	$\kappa_3^4\boxed{0}$	$\kappa_4^4\boxed{0}$
q_5	$\kappa_1^5\boxed{0}$	$\kappa_2^5\boxed{0}$	$\kappa_3^5\boxed{0}$	$\kappa_4^5\boxed{0}$
q_6	$\kappa_1^6\boxed{0}$	$\kappa_2^6\boxed{0}$	$\kappa_3^6\boxed{1}$	$\kappa_4^6\boxed{0}$
q_7	$\kappa_1^7\boxed{0}$	$\kappa_2^7\boxed{0}$	$\kappa_3^7\boxed{1}$	$\kappa_4^7\boxed{0}$

D. PROBLEM DEFINITION

In our study, we focus on the problem of scheduling and routing. This problem can be outlined as follows: We explore a system based on Time-Sensitive Networking (TSN), consisting of an array of bridges. These bridges, in turn, connect to multiple end-systems. Each end-system houses one or more tasks. Upon completion of these tasks, data streams are generated; these streams then emanate from the talkers and initiate tasks once they reach the listeners.

Within this intricate network, every task and its corresponding streams — collectively termed an 'application' — function in harmony. Synchronizing these applications is crucial for achieving network performance hallmarked by minimal latency and reasonable response time. This desired state can be reached through precise task scheduling within the end-systems. Moreover, it requires choosing the best route and schedule for data streams traversing the bridges, all the while upholding the network's integrity to avert any possible data frame losses.

1) EXEMPLARY PROBLEM INSTANCE

To obtain a clearer understanding of the problem, refer to Fig. 2, which depicts a network comprising nodes $\mathcal{V}$ and links $\mathcal{E}$. The nodes are represented as $\mathcal{V} = \{es_1, es_2, es_3, es_4, br_1, br_2, br_3, br_4\}$, and the links between these nodes are denoted as $\mathcal{E}$, which include the following pairs: $\mathcal{E} = \{(es_1, br_1), (es_1, br_2), (es_2, br_1), (es_2, br_2), (es_3, br_3), (es_3, br_4), (es_4, br_3), (es_4, br_4), (br_1, br_2), (br_1, br_3), (br_1, br_4), (br_1, es_1), (br_1, es_2), (br_2, br_1), (br_2, br_3), (br_2, br_4), (br_2, es_1), (br_2, es_2), (br_3, br_1), (br_3, br_2), (br_3, br_4), (br_3, es_3), (br_3, es_4), (br_4, br_1), (br_4, br_2), (br_4, br_3), (br_4, es_3), (br_4, es_4)\}$.

For every link (υ_a, υ_b)in $\mathcal{E}$, we assign a link speed of $\Omega_{(\upsilon_a,\upsilon_b)}$= 10Mbps and a propagation delay set to zero. Each end-system engaged in the transmission or receipt of streams will execute a task or a series of tasks related to that particular stream. These tasks, together with their corresponding

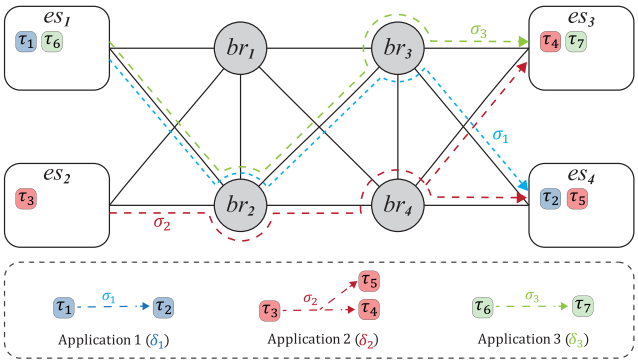


FIGURE 2. The network topology of the exemplary problem instance illustrates the route path taken by each stream.

streams, should be integrated within a single application, either autonomously or in combination with other tasks and streams.

The set of applications is denoted by $\Delta = \{\delta_1, \delta_2, \delta_3\}$. Each application comprises one stream: δ_1 and δ_3 both contain two tasks, while δ_2 encompasses three tasks. Tasks from the same application reside at both the talker and listener end-systems, connected by a single stream.

Each stream in $S = \{\sigma_1, \sigma_2, \sigma_3\}$ has a specific size. These sizes, represented by L_1^σ, L_2^σ, and L_3^σ are 70*Bytes*, 80*Bytes*, and 35*Bytes*, respectively. The worst-case execution time (WCET) ω_j^τ for tasks in δ_1 and δ_2 is 30μs. In contrast, tasks within δ_3 have a ω_j^τ of 20μs.

There are two distinct traffic types with varied priorities. Streams from δ_1 and δ_2 rely on a Time-Triggered (TT) protocol, while the stream from δ_3 operates on a Best-Effort (BE) basis, as delineated in Table 3. The periods assigned to tasks and streams align with the periods of their corresponding applications. For simplicity in this example, we set a consistent period and deadline of 1000μs across all applications. As such, the Hyperperiod (HP) for our applications is derived as $HP = \text{lcm}\{\delta_1, \delta_2, \delta_3\} = \text{lcm}\{1000, 1000, 1000\} = 1000\mu s$.

TABLE 3. A Queue-Gating Time (QGT) determines when each queue is open (indicated by a status of "1") or closed (indicated by a status of "0") at a particular gating time.

Δ	Γ_k^δ	T	ω_j^τ	Γ_j^τ	S	ρ_i^σ	L_i^σ	Γ_i^σ
δ_1	1000	τ_1, τ_2	30	1000	σ_1	TT	70	1000
δ_2	1000	τ_3, τ_4, τ_5	30	1000	σ_2	TT	80	1000
δ_3	1000	τ_6, τ_7	20	1000	σ_3	BE	35	1000

Each of the four bridges is equipped with eight queues, each with a corresponding gate. The first two queues, q_0 and q_1, are designated for TT traffic. In contrast, the last two queues, q_6 and q_7, cater to BE traffic. The middle queues—q_2, q_3, q_4, and q_5—remain unused.

We have defined a series of gating times t_φ, culminating in a final gating time of 1000μs. Specifically, the first is at 0μs, the second at 400μs, the third at 700μs, with the last precisely at 1000μs. This timing aligns with the period and deadline set for our applications. The gates g_0 and g_1, associated with the first two queues, enter the "open status" phase $\psi_0 = \psi_1 = 1$ at 0μs and switch to the "closed status" phase $\psi_0 = \psi_1 = 0$ at 400μs. This interval is specifically for processing the TT traffic in these queues.

As detailed in Table 3, the gates g_6 and g_7 begin in the "closed status" $\psi_6 = \psi_7 = 0$ at 1000μs before transitioning to the "open status" $\psi_6 = \psi_7 = 1$ at 700μs. This ensures uninterrupted TT traffic transmission across the bridges without interference from BE traffic. The subsequent 1000μs period is reserved for BE traffic, ensuring the reliable delivery of TT traffic.

Based on the GCL entry presented in Table 4, the gates g_6 and g_7 maintain a "closed status" $\psi_6 = \psi_7 = 0$ for the initial 700μs. This ensures the seamless transmission of TT traffic across the bridges, free from interference from BE traffic. At 700μs, these gates transition to an "open status" $\psi_6 = \psi_7 = 1$, lasting until they return to a "closed status" at 1000μs. This period is reserved for BE traffic processing. Fig. 3 showcases the configuration of gates and queues, detailing time allocations for each traffic type. As specified in the GCL entry in Table 4, and combined with the QGT $\kappa_\varphi^q \boxed{\psi}$ calculations detailed in Section III-C, the $\kappa_\varphi^q \boxed{\psi}$ for our example is elaborated in Table 5. QGT indicates when each queue is either open (marked by a status of "1") or closed (marked by a status of "0") relative to a specific gating time.

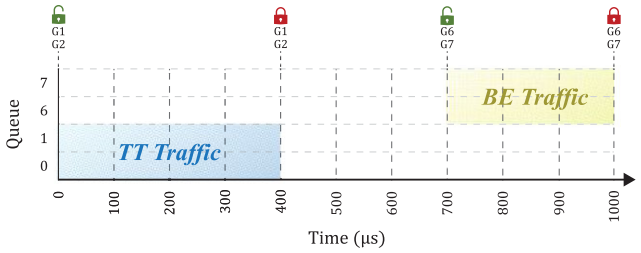


FIGURE 3. The opening and closing times of the gates for each queue, along with the type of traffic permitted for transmission during those open intervals.

TABLE 4. The gate control list (GCL) entry based on different gating times.

t_φ	GCL entry
0μs	00000011
400μs	00000000
700μs	11000000
1000μs	00000000

In the subsequent example and throughout this text, we use the microsecond (μs) and millisecond (ms) as our primary time units. Fig. 4 presents the schedule of our example via a Gantt chart. This visualization displays tasks and streams concurrently, with varied colors distinguishing the specific applications to which they correspond. Shaded blocks highlight the periods when distinct traffic types have transmission

TABLE 5. The values of the Queue-Gating Time (QGT) for each queue at a particular gating time.

$\kappa_\varphi^q[\psi]$	$t_1 = 0\mu s$	$t_2 = 400\mu s$	$t_3 = 700\mu s$	$t_4 = 1000\mu s$
q_0	$\kappa_1^0[1] = 0\mu s$	$\kappa_2^0[0] = 400\mu s$	$\kappa_3^0[0] = 700\mu s$	$\kappa_4^0[0] = 1000\mu s$
q_1	$\kappa_1^1[1] = 0\mu s$	$\kappa_2^1[0] = 400\mu s$	$\kappa_3^1[0] = 700\mu s$	$\kappa_4^1[0] = 1000\mu s$
q_2	$\kappa_1^2[0] = 0\mu s$	$\kappa_2^2[0] = 400\mu s$	$\kappa_3^2[0] = 700\mu s$	$\kappa_4^2[0] = 1000\mu s$
q_3	$\kappa_1^3[0] = 0\mu s$	$\kappa_2^3[0] = 400\mu s$	$\kappa_3^3[0] = 700\mu s$	$\kappa_4^3[0] = 1000\mu s$
q_4	$\kappa_1^4[0] = 0\mu s$	$\kappa_2^4[0] = 400\mu s$	$\kappa_3^4[0] = 700\mu s$	$\kappa_4^4[0] = 1000\mu s$
q_5	$\kappa_1^5[0] = 0\mu s$	$\kappa_2^5[0] = 400\mu s$	$\kappa_3^5[0] = 700\mu s$	$\kappa_4^5[0] = 1000\mu s$
q_6	$\kappa_1^6[0] = 0\mu s$	$\kappa_2^6[0] = 400\mu s$	$\kappa_3^6[1] = 700\mu s$	$\kappa_4^6[0] = 1000\mu s$
q_7	$\kappa_1^7[0] = 0\mu s$	$\kappa_2^7[0] = 400\mu s$	$\kappa_3^7[1] = 700\mu s$	$\kappa_4^7[0] = 1000\mu s$

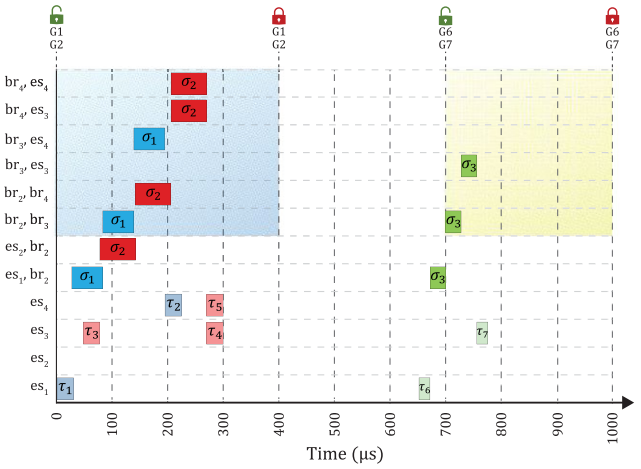


FIGURE 4. The network schedule of the exemplary problem instance for the tasks and streams.

permission across each bridge, influenced by the gate statuses. In this context, the Hyperperiod (HP) is $1000\mu s$. The bottom four rows showcase end-systems where tasks commence and their corresponding streams arise. Tasks interlinked within the same application bear the same color. The remaining ten rows detail the links between device pairs, marking frame occurrences.

For instance, the stream, labeled as σ_1 and colored light blue, emerges first at the link between es_1 and br_2. It initiates transmission at $30\mu s$, aligning with the end time of task τ_1, and culminates at $86\mu s$. To discern the stream's duration, we divide its size ($70Bytes$) by the link speed ($10Mbps$), resulting in $56\mu s$ for σ_1. It is pivotal to acknowledge that we determine durations of other streams similarly, factoring in the frame overhead within the stream size.

Rectangles shaded in halftone blue denote gate opening times for the TT queues, while those in halftone yellow symbolize gate opening times for BE queues. On these rectangles, the x-axis depicts gate operation timings, while the y-axis signifies the bridges serving as conduits for the traversing streams.

IV. OPTIMIZATION FRAMEWORK

The optimization problems similar to the one highlighted in this paper are classified as NP-hard due to their potential reduction to the Bin-Packing problem, as referenced in [19] and [25]. When dealing with large input data sizes, these problems can become intractable. To find solutions, we employ Constraint Programming (CP) in our optimization framework. We have implemented an approach called "Optimized Hybrid Deterministic Scheduling and Routing" (OHDSR). The term "hybrid" refers to the combination of two types of traffic: TT and BE. This optimization framework efficiently finds robust solutions within a reasonable time frame.

A. TIME-SENSITIVE CONSTRAINTS

The aim of any optimization solver is to identify the most suitable solution from a range of possible solutions to a problem. This selection is determined by specific criteria defined by an objective function. The optimization process involves adhering to several constraints. These constraints serve as boundaries or conditions that describe the acceptable solutions within the optimization problem. In this section, we will discuss the constraints we implemented and categorize them into four sub-sections: routing constraints, priority scheduling constraints, stream scheduling constraints, and task scheduling constraints.

1) ROUTING CONSTRAINTS

To transmit a stream, denoted as σ_i, from a source node (referred to as the talker) across a network – passing through various intermediate nodes (known as bridges) – and eventually reaching the destination nodes (termed listeners), certain routing constraints must be established. The constraints detailed in this section are influenced by the work presented in [25] and [34].

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: $\{\upsilon_a\} \cup \{\upsilon_b \in \mathcal{V}\ where\ (\upsilon_a, \upsilon_b) \in \mathcal{E}\} \cup \{nil\}$.

For every stream in the network, $Z_{\upsilon_a}^{\sigma_i}$ will be calculated for each node within the network, taking into account all bridges and end-systems. In case of υ_a is the talker, then $Z_{\upsilon_a}^{\sigma_i} = \upsilon_a$. If υ_a lies along the stream's path, then $Z_{\upsilon_a}^{\sigma_i} = \upsilon_b$. Conversely, if υ_a is not on the path of the stream, $Z_{\upsilon_a}^{\sigma_i} = nil$.

Referring to the exemplary instance in section III-D-I, and considering stream σ_1, its path is described as: $[es_1 \longrightarrow br_2 \longrightarrow br_3 \longrightarrow es_4]$ Consequently, $Z_{\upsilon_a}^{\sigma_1}$ will be es_1 if $\upsilon_a = es_1$ and if $\upsilon_a = br_2$, while $Z_{\upsilon_a}^{\sigma_1}$ will equal br_2 if $\upsilon_a = br_3$, and br_3 if $\upsilon_a = es_4$. In the case where υ_a is any of the nodes not on the path of the stream σ_1 (like es_2, es_3, br_1, and br_4), $Z_{\upsilon_a}^{\sigma_i} = nil$ for all these node, Fig. 5 clarify the concept behind this approach. Additionally, the length of the path from υ_a to the talker-node of stream σ_i is denoted as $\Upsilon_{\upsilon_a}^{\sigma_i}$, with the allowable range of values is defined as: $\{0 \leq \Upsilon_{\upsilon_a}^{\sigma_i}, \leq |BR| + 1\}$. The routing constraints are outlined in equations 1

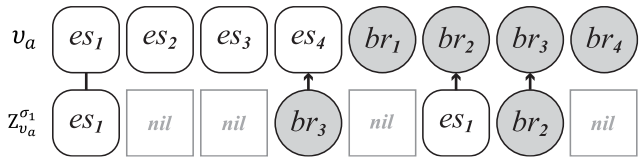


FIGURE 5. The value of $Z^{\sigma_1}_{\upsilon_a}$ for Stream σ_1 from the exemplary problem instance.

through 5, as follows:

$$\forall \sigma_i \in S, \forall v_a \in \mathcal{V} except \left\{es\left\langle\tau^{\sigma_i}_{talker}\right\rangle\right\} : \left(Z^{\sigma_i}_{v_a} \neq nil\right) \Rightarrow \left(\Upsilon^{\sigma_i}_{v_a} = \Upsilon_{z^{\sigma_i}_{v_a}} + 1\right) \tag{1}$$

Equation (1) demonstrate that, for each stream, if the successor of node υ_a is not equivalent to nil, indicating the presence of a successor for node υ_a along the path of stream σ_i, then the length of the path to node υ_a should equal the length of the path to its successor node $Z^{\sigma_i}_{\upsilon_a}$, increased by one. This constraint helps in avoiding cycles in the route of the stream.

$$\forall \sigma_i \in S, \quad \forall \upsilon_a, \upsilon_b \in \mathcal{V} : Z^{\sigma_i}_{\upsilon_b} = nil \Longleftrightarrow Z^{\sigma_i}_{\upsilon_a} \neq \upsilon_b \tag{2}$$

Equation (2) illustrate that each node with a successor along the path of stream σ_i must also have a predecessor, and vice versa, considering the reverse route used. This helps in preventing any loose ends.

$$\forall \sigma_i \in S, \quad \forall v_a \in es\left\langle\tau^{\sigma_i}_{listener}\right\rangle \text{ , where } \tau^{\sigma_i}_{listener} \in T^{\sigma_i}_{listener} : Z^{\sigma_i}_{v_a} \neq nil \tag{3}$$

Equation (3) states that every listener end-system of stream σ_i must have a successor.

$$\forall \sigma_i \in S : Z^{\sigma_i}_{es\left\langle\tau^{\sigma_i}_{talker}\right\rangle} = es\left\langle\tau^{\sigma_i}_{talker}\right\rangle, \Upsilon^{\sigma_i}_{es\langle\tau_{talker}\rangle} = 0 \tag{4}$$

Equation (4) impose that for every stream σ_i, the talker-node $es\left\langle\tau^{\sigma_i}_{talker}\right\rangle$, identifies itself as the successor, with the respective path length being zero at the talker's position.

$$\forall \sigma_i \in S, \forall v_a \in ES \; except \left\{es\left\langle\tau^{\sigma_i}_{listener}\right\rangle and \; es\left\langle\tau^{\sigma_i}_{talker}\right\rangle\right\} where \; \tau^{\sigma_i}_{listener} \in T^{\sigma_i}_{listener} : Z^{\sigma_i}_{v_a} = nil \tag{5}$$

Equation (5) indicates that the successor node will be nil for any end-systems in the network that do not possess a listener-node task $\tau^{\sigma_i}_{listener}$ or talker-node task $\tau^{\sigma_i}_{talker}$. It is important to note that this applies only to end-system nodes, excluding the bridge nodes.

2) PRIORITY SCHEDULING CONSTRAINTS

The priority scheduling constraint is applied to each bridge $br_n \in BR$, which accommodates streams $\sigma_i \in S$ of various types ρ^{σ}_i, passing through it. As discussed in section III-D, each bridge is equipped with a set of eight queues $q^{br_n}_{\phi} \in Q^{br_n}$, each characterized by different types ρ^{q}_{ϕ}. Streams sharing the type with a specific queue will be allocated to that queue.

Every queue operates with two QGTs $t^{q}_{\varphi}\boxed{\psi}$, governing both the open (1) and closed (0) statues. These times are contingent upon the gating time $\mathbb{t}^{br_n}_{\varphi} \in \mathbb{T}^{br_n}$ associated with each bridge, from which the respective open and closed times are calculated.

$$\forall \sigma_i \in S, \quad \forall br_n \in BR, \quad \forall q^{br_n}_{\varphi} \in Q^{br_n}, \forall \mathbb{t}^{br_n}_{\varphi} \in \mathbb{T}^{br_n}, \quad \rho^{\sigma}_i == \rho^{q}_{\phi} : \eta^{\sigma_i}_{br_n} \geq t^{q^{br_n}}_{\varphi}\boxed{1} \; \zeta^{\sigma_i}_{br_n} \leq t^{q^{br_n}}_{\varphi}\boxed{0} \tag{6}$$

Equation (6) assigns the time domain for both the offset and the end-time of each stream at each bridge. The offset of stream σ_i should be greater than or equal to the QGT when the status is 1, indicating "open" status. Conversely, the end-time of that stream must be less than or equal to the QGT when the status is 0, denoting "closed" status. This constraint will not apply to the offsets and the end-times of streams at end-systems $es_m \in ES$, and at links $(\upsilon_a, \upsilon_b) \in \mathcal{E}$, as the queues and gates are exclusively incorporated in the design of the bridges.

3) STREAM SCHEDULING CONSTRAINTS

In this section, we will introduce the stream scheduling constraint. Adhering to this constraint can potentially facilitate the optimal solution within our optimization framework by finding the best achievable schedule. The constraints are as follows:

$$\forall \sigma_i \in S, \; \forall (\upsilon_a, \; \upsilon_b) \in \mathcal{E}, \quad \forall (\upsilon_a, \; \upsilon_b) \in \mathbb{R}^{\sigma}_i : \varepsilon^{\sigma_i}_{(\upsilon_a, \; \upsilon_b)} = \left\lceil \frac{L^{\sigma}_i}{\Omega_{(\upsilon_a, \; \upsilon_b)}} \right\rceil \tag{7}$$

The first constraint in the process of scheduling streams is determining the duration time for each stream σ_i at each link (υ_a, υ_b). Equation (7) depicts this by calculating the time as the division of the stream size (in Bytes) by the link speed (in Mbps), thereby yielding a duration measured in microseconds (μs).

$$\forall \sigma_i \in S, \; \forall (\upsilon_a, \; \upsilon_b) \in \mathcal{E}, \quad \forall (\upsilon_a, \; \upsilon_b) \in \mathbb{R}^{\sigma}_i : \zeta^{\sigma}_{(\upsilon_a, \; \upsilon_b)} = \eta^{\sigma}_{(\upsilon_a, \; \upsilon_b)} + \varepsilon^{\sigma}_{(\upsilon_a, \; \upsilon_b)} \tag{8}$$

After assigning the duration to each stream, we stipulate in (8) that the end time of each stream should be equal to the sum of that stream's offset and its duration. This constraint is applied to all streams $\sigma_i \in S$ across all links on their respective paths $\forall (\upsilon_a, \upsilon_b) \in \mathbb{R}^{\sigma}_i$.

$$\forall \sigma_i \in S, \; \forall (\upsilon_b, \; \upsilon_c) \in \mathcal{E} \cap \mathbb{R}^{\sigma}_i, \quad \upsilon_a = Z^{\sigma}_{\upsilon_b} : \zeta^{\sigma}_{(\upsilon_a, \; \upsilon_b)} \leq \eta^{\sigma}_{(\upsilon_b, \; \upsilon_c)} \tag{9}$$

Equation (9) enforces the precise sequencing of any two links that are located along the route path R^{σ}_i of any stream σ_i.

This involves incorporating the links υ_a, υ_b and υ_b, υ_c as parts of this route. Given that this stream progresses through the link υ_a, υ_b before advancing to υ_b, υ_c, the offset of that stream at the link υ_b, υ_c must be greater than or equal to the end time of the preceding link υ_a, υ_b.

$$\begin{aligned}
&\forall \sigma_1, \sigma_2 \in S,\ \sigma_1 \neq \sigma_2,\ \forall (\upsilon_a,\ \upsilon_b) \in \mathbb{R}_1^\sigma \cap \mathbb{R}_2^\sigma, \\
&\quad \forall \alpha \in \left\{0,\ \cdots,\ \frac{lcm\left(\Gamma_1^\sigma, \Gamma_2^\sigma\right)}{\Gamma_1^\sigma}\right\}, \\
&\quad \forall \beta \in \left\{0,\ \cdots,\ \frac{lcm\left(\Gamma_1^\sigma, \Gamma_2^\sigma\right)}{\Gamma_2^\sigma}\right\} : \\
&\quad \left(\left(\alpha \times \Gamma_1^\sigma + \zeta_{(\upsilon_a,\ \upsilon_b)}^{\sigma_1}\right) \leq \left(\beta \times \Gamma_2^\sigma + \eta_{(\upsilon_a,\ \upsilon_b)}^{\sigma_2}\right)\right) \vee \\
&\quad \left(\left(\beta \times \Gamma_2^\sigma + \zeta_{(\upsilon_a,\ \upsilon_b)}^{\sigma_2}\right) \leq \left(\alpha \times \Gamma_1^\sigma + \eta_{(\upsilon_a,\ \upsilon_b)}^{\sigma_1}\right)\right)
\end{aligned} \tag{10}$$

Equation (10) helps to prevent any overlap between any two different streams σ_1 and σ_2, which share a link (υ_a, υ_b) within their respective routes $\mathbb{R}_1^\sigma$ and $\mathbb{R}_2^\sigma$. This entails that the timeframe during which stream σ_1 utilizes the link should not coincide with that of stream σ_2. Implementing this constraint ensures equitable resource allocation, thereby enhancing both the efficiency and predictability of the scheduling process.

$$\begin{aligned}
&\forall \sigma_1, \sigma_2 \in S,\ \sigma_1 \neq \sigma_2,\ \forall (\upsilon_b,\ \upsilon_c) \in \mathbb{R}_1^\sigma \cap \mathbb{R}_2^\sigma, \\
&\quad \forall (\upsilon_{a1},\ \upsilon_b) \in \mathbb{R}_1^\sigma,\ \ \forall (\upsilon_{a2},\ \upsilon_b) \in \mathbb{R}_2^\sigma, \\
&\quad \upsilon_{a1} = Z_{\upsilon_b}^{\sigma_1},\quad \upsilon_{a2} = Z_{\upsilon_b}^{\sigma_2}, \\
&\quad \forall \alpha \in \left\{0,\ \cdots,\ \frac{lcm\left(\Gamma_1^\sigma, \Gamma_2^\sigma\right)}{\Gamma_1^\sigma}\right\}, \\
&\quad \forall \beta \in \left\{0,\ \cdots,\ \frac{lcm\left(\Gamma_1^\sigma, \Gamma_2^\sigma\right)}{\Gamma_2^\sigma}\right\} : \\
&\quad \left(\left(\alpha \times \Gamma_2^\sigma + \eta_{(\upsilon_b,\ \upsilon_c)}^{\sigma_2}\right) \leq \left(\beta \times \Gamma_1^\sigma + \eta_{(\upsilon_{a1},\ \upsilon_b)}^{\sigma_1}\right)\right) \vee \\
&\quad \left(\left(\beta \times \Gamma_1^\sigma + \eta_{(\upsilon_b,\ \upsilon_c)}^{\sigma_1}\right) \leq \left(\alpha \times \Gamma_2^\sigma + \eta_{(\upsilon_{a2},\ \upsilon_b)}^{\sigma_2}\right)\right)
\end{aligned} \tag{11}$$

To enhance the resilience of frame transmissions and prevent any delays or loss of frames, Equation (11) is introduced. At the egress port of each node υ_b (which, in this case, should function as bridge br_n), it is notable that this bridge node is a part of the link (υ_b, υ_c). The link (υ_b, υ_c) is located on the routes $\mathbb{R}_1^\sigma$ and $\mathbb{R}_1^\sigma$ common to both streams σ_1 and σ_2. Meanwhile, the link $(\upsilon_{a1}, \upsilon_b)$ is part of the route $\mathbb{R}_1^\sigma$, and the link $(\upsilon_{a2}, \upsilon_b)$ is found on the route R_2^σ. The two frames arriving at the bridge node υ_b came from different sources. The first frame arrives from node υ_{a1} in the stream σ_1 via the link $(\upsilon_{a1}, \upsilon_b)$, while the second frame arrives from node υ_{a2} in the other stream σ_2 via the link $(\upsilon_{a2}, \upsilon_b)$. Moreover, the frames arriving at the ingress port of node υ_b should not overlap within the time domain. This principle of frame isolation, originally suggested by Craciunas et al. [35], has become a widely accepted practice in configuring TSN networks.

4) TASK SCHEDULING CONSTRAINTS

Tasks are vital components of applications and are linked to various streams within those applications. Establishing an optimal schedule for these tasks is important. To achieve this, the following constraints should be fulfilled:

$$\forall \tau_j \in T : \zeta_{\upsilon_a}^\tau = \eta_{\upsilon_a}^\tau + \omega_j^\tau \tag{12}$$

Equation (12) determines the end-time of each task τ_j at the node υ_a where this task is executed. Each task possesses a worst-case execution time ω_j^τ, representing the duration required for the task to complete. The end-time of a task is calculated by adding its execution time to its offset time.

$$\begin{aligned}
&\forall \tau_j \in T,\quad \forall \sigma_i \in S,\quad \tau_j = \tau_{talker}^{\sigma_i}, \\
&\qquad \forall (\upsilon_a,\ \upsilon_b) \in \mathcal{E} \cap \mathbb{R}_i^\sigma,\quad \upsilon_a == es_j^\tau : \\
&\qquad \zeta_{\upsilon_a}^\tau \leq \eta_{(\upsilon_a,\ \upsilon_b)}^\sigma
\end{aligned} \tag{13}$$

As we highlighted before, certain tasks and streams within the same application are interconnected. The coordination between these interconnected tasks and streams needs to be carefully planned with regard to their timing sequences. As depicted in Equation (13), the tasks taking place at the talker nodes need to be initiated and completed before the beginning of the corresponding outgoing stream. This implies that the offset of stream σ_i, originating from node υ_a and passing through the link (υ_a, υ_b), should be greater than or equal to the completion time of task τ_j, which is executed at node υ_a.

$$\begin{aligned}
&\forall \tau_j \in T,\quad \forall \sigma_i \in S,\quad \tau \in T_{listner}^{\sigma_i}, \\
&\qquad \forall (\upsilon_a,\ \upsilon_b) \in \mathcal{E} \cap \mathbb{R}_i^\sigma,\quad \upsilon_b == es_j^\tau : \\
&\qquad \zeta_{(\upsilon_a,\ \upsilon_b)}^\sigma \leq \eta^\tau
\end{aligned} \tag{14}$$

Building upon the principles presented in (13), Equation (14) addresses the process concerning the incoming streams at the listener nodes and the corresponding tasks executed at these nodes. It is essential that tasks at any listener node are not initiated prior to the receipt of the corresponding incoming stream. This means that the offset of task τ_j located at node υ_b must greater than or equal to the end-time of stream σ_i reaching node υ_b and transiting through the link (υ_a, υ_b).

$$\begin{aligned}
&\forall \tau_1, \tau_2 \in T, \tau_1 \neq \tau_2, \\
&\quad \forall \tau_1, \tau_2 \in T,\ \ \tau_1 \neq \tau_2, \\
&\qquad \forall \alpha \in \left\{0,\ \cdots,\ \frac{lcm\left(\Gamma_1^\tau, \Gamma_2^\tau\right)}{\Gamma_1^\tau}\right\}, \\
&\qquad \forall \beta \in \left\{0,\ \cdots,\ \frac{lcm\left(\Gamma_1^\tau, \Gamma_2^\tau\right)}{\Gamma_2^\tau}\right\}, \\
&\qquad \left(\left(\alpha \times \Gamma_1^\tau + \zeta_{\upsilon_a}^{\tau_1}\right) \leq \left(\beta \times \Gamma_2^\tau + \eta_{\upsilon_a}^{\tau_2}\right)\right) \vee \\
&\qquad \left(\left(\beta \times \Gamma_2^\tau + \zeta_{\upsilon_a}^{\tau_2}\right) \leq \left(\alpha \times \Gamma_1^\tau + \eta_{\upsilon_a}^{\tau_1}\right)\right)
\end{aligned} \tag{15}$$

Equation (15) guarantees that no overlap occurs between any two different tasks τ_1 and τ_2, when executed at the same end-system υ_a.

B. OBJECTIVE FUNCTIONS

Within the structure of our optimization framework, we focus on stream routing and the scheduling of both streams and tasks, while taking into account the priorities associated with the different types of these streams. To facilitate this, we introduce two objective functions in this section: one related to routing and the other concerning scheduling.

1) ROUTING OBJECTIVE FUNCTION

The objective of routing optimization explored in this paper is to minimize the total sum of the lengths of the routes of all streams, as depicted in (16). For each node, if the successor node is not *nil*, it will be included in the total nodes accounted for the stream σ_i, excluding the originating talker node. The values obtained are then summed for each stream and subsequently minimized.

$$min \sum_{\sigma_i \in S} \sum_{\forall v_a \in \mathcal{V} \; except \; \{es\left\langle \tau_{talker}^{\sigma_i} \right\rangle\}} (Z_{v_a}^{\sigma_i} != nil) \quad (16)$$

2) SCHEDULING OBJECTIVE FUNCTION

In our study, the objective of scheduling optimization is to minimize the sum of the latencies for all transmissions, as delineated in (17). As illustrated in Fig. 6, latency refers to the duration of a transmission, beginning at the initiation point (offset) of the task τ_{talker} at talker node $es^{\tau_{talker}}$ and taking into account the time required for stream transmission until the completion (end-time) of the task $\tau_{listner}$ at talker node $es^{\tau_{listener}}$.

$$min \sum_{\delta_k \in \Delta} \left(\zeta^{\tau_{listner}} - \eta^{\tau_{talker}}\right), \quad where \;\; \tau_{listner}, \;\; \tau_{talker} \in T_k \quad (17)$$

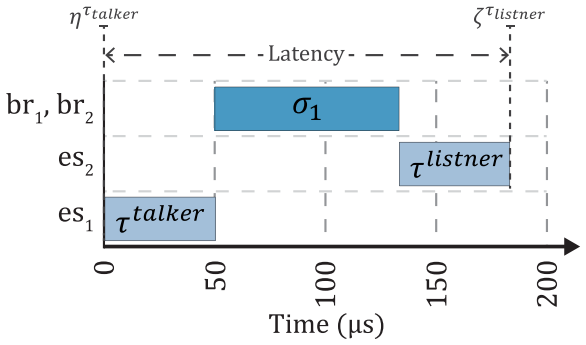


FIGURE 6. The value of $Z_{v_a}^{\sigma_i}$ for Stream σ_1 from the exemplary problem instance.

Each application comprises multiple tasks that are executed at the end-systems, with streams being transferred from one end-system to another via bridges. Every task and stream have a start-time, end-time, and duration, which represent the interval between the initiation and ending points. Through the implementation of this objective, we aim to significantly reduce the total latency, thereby fulfilling one of the primary targets in time-sensitive networks.

V. EXPERIMENTAL EVALUATION

In this section, we assess the performance of our optimization framework using various synthetic problem instances through a setup compatible with our framework.

A. PROBLEM INSTANCES AND EXPERIMENTAL SETUP

For a comprehensive evaluation of the proposed framework, we chose a variety of problem instances representative of characteristics commonly found in realistic automation and autonomous vehicles networks. We categorized these instances into four groups based on the number of bridges used: 6, 12, 18, and 24. Each group comprises six problem instances. These are further divided into two subgroups, each with distinct numbers of end-systems. The first subgroup has a 1.5:1 ratio of end-systems to bridges, while the second has a 2:1 ratio, as detailed in Table 6.

TABLE 6. Problem instance.

Instance Name	No. BRs	No. ESs	No. Tasks	No. TT Streams	No. BE Streams
Exemplary	4	4	7	2	1
6-9-A	6	9	43	6	12
6-9-B	6	9	43	9	9
6-9-C	6	9	45	12	6
6-12-A	6	12	56	8	16
6-12-B	6	12	57	12	12
6-12-C	6	12	56	16	8
12-18-A	12	18	84	12	24
12-18-B	12	18	87	18	18
12-18-C	12	18	84	24	12
12-24-A	12	24	116	16	32
12-24-B	12	24	112	24	24
12-24-C	12	24	113	32	16
18-27-A	18	27	128	18	36
18-27-B	18	27	129	27	27
18-27-C	18	27	130	36	18
18-36-A	18	36	169	24	48
18-36-B	18	36	165	36	36
18-36-C	18	36	172	48	24
24-36-A	24	36	173	24	48
24-36-B	24	36	171	36	36
24-36-C	24	36	176	48	24
24-48-A	24	48	227	32	64
24-48-B	24	48	223	48	48
24-48-C	24	48	224	64	32

Within this setup, connectivity is established through bi-directional, full-duplex links. These links connect both end-systems to bridges and bridges amongst themselves, operating at a transmission speed of 1 Gbps. In our research, we devised an algorithm to construct network topologies for the specified problem instances. To implement this algorithm, we utilized the NetworkX Python library [36], which aids in establishing graphs that depict the network topologies. Bridge edges connecting to other bridges are chosen at random. We've tweaked the algorithm to ensure that end-systems don't establish direct links with one another.

Furthermore, every end-system connects to a minimum of two and a maximum of four bridges, selected at random.

To further diversify our testing scenarios, we vary the data stream configurations for each subgroup. The first subgroup mainly uses TT streams over BE streams, the second has an equal number of both, while the third primarily employs BE streams. This methodology allows for a comprehensive evaluation of our framework across diverse scenarios within a consistent network topology. Notably, around 1:4 of the streams within the same application will be multicast, irrespective of their priorities.

Additionally, the number of tasks ranges from 43 to 224, influenced by the number of streams in each application. These tasks are assigned to the end-systems and display random worst-case execution times, which can be as much as three percent of their designated periods. Finally, we chose to cap stream sizes at figures below the maximum transmission unit of 1500*Bytes*, selecting sizes at random with the frame overhead taken into account.

Each application has a period set at $1000\mu s$, ensuring a consistent period across all its streams and tasks. Every bridge comes with eight queues. Notably, the first two queues are designated for TT streams, while the last two are reserved for BE streams. The middle four queues remain unused.

Each queue features a gate that functions based on a predetermined gating time specified in the GCL. This gating time is set by the network's application Hyperperiod under evaluation. The gates for the TT queues open right at the start of the Hyperperiod, always at 0μs. They closed after 50%-60% of the Hyperperiod has elapsed. Following this, all gates stay closed for a duration ranging from 0% to 5% of the Hyperperiod. The remaining time within the Hyperperiod is reserved for opening the BE queue gates.

Constraint programming (CP) is widely used, with many tools compatible across different programming languages to tackle CP optimization challenges. In our study, we employed the CP-Sat Solver [37], a component of the OR-Tools suite—an open-source collection developed by Google. We tested all problem instances on a system powered by an Intel Core i7-11370H CPU, running at 3.30 GHz with 8 CPUs and 8GB of memory.

B. NUMERICAL RESULTS AND DISCUSSION

In this subsection, we provide a detailed review of our results for each problem instance. We begin by evaluating the latency, followed by an analysis of the response times. Finally, we compare our response time results with findings from previous related studies.

1) LATENCY EVALUATION

First, we aim to evaluate the overall latency of the transmission. Latency is defined as the time taken from the start of the sender task to the end-time of the last receiver task. Additionally, we evaluate the worst-case latency for both TT and BE streams to guarantee that the latency doesn't impact their reception times.

In Table 7, instances are highlighted where the number of BE streams surpasses that of TT streams. We refer to instances of this scenario as BE-dominant. Despite this, TT traffic retains its top priority. The total latency increased rapidly with larger scale instances having a higher count of streams. Due to the larger number of BE streams, the worst-case latency for BE streams is typically higher than that for TT streams in most scenarios. Among all instances, the 12-18-A instance exhibits the greatest difference with a maximum of 14.29%, indicating a TT-dominant advantage.

TABLE 7. The total latency, the worst-case latency for Time-Triggered (TT) streams, and the worst-case latency for Best Effort (BE) streams, for the BE-dominant problem instances.

Instance Name	Total Latency (µs)	Worst-Case Latency of TT streams (µs)	Worst-Case Latency of BE streams (µs)
6-9-A	917	57	61
6-12-A	1313	63	62
12-18-A	1910	56	64
12-24-A	2553	62	62
18-27-A	2896	65	62
18-36-A	3862	64	64
24-36-A	3942	62	64
24-48-A	5172	64	65

Table 8 addresses instances where the number of TT and BE streams are equal. For this balanced TT-BE scenario, the worst-case latency values are closely matched, indicating an even distribution of the load between the two stream types. The total latency rises based on the scale of the instances. This latency pattern is similar to what we saw with the BE-dominant scenarios mentioned earlier, and it looks like the TT-dominant scenarios in Table 9.

TABLE 8. The total latency, the worst-case latency for Time-Triggered (TT) streams, and the worst-case latency for Best Effort (BE) streams, for the balanced TT-BE problem instances.

Instance Name	Total Latency (µs)	Worst-Case Latency of TT streams (µs)	Worst-Case Latency of BE streams (µs)
6-9-B	948	60	60
6-12-B	1277	59	62
12-18-B	1942	65	60
12-24-B	2538	59	63
18-27-B	2944	64	62
18-36-B	3927	68	62
24-36-B	3929	66	62
24-48-B	5270	65	66

An observation from Table 9 is that even when TT streams are dominant in the final set of instances, the variance in worst-case latency is small. This suggests that our model consistently delivers stable performance for BE streams, even when paired with the higher-priority TT streams. Such observations provide deeper insight into the system's adeptness at managing varying stream priorities.

TABLE 9. The total latency, the worst-case latency for Time-Triggered (TT) streams, and the worst-case latency for Best Effort (BE) streams, for the TT-dominant problem instances.

Instance Name	Total Latency (µs)	Worst-Case Latency of TT streams (µs)	Worst-Case Latency of BE streams (µs)
6-9-C	966	60	58
6-12-C	1228	58	59
12-18-C	1926	60	63
12-24-C	2638	63	60
18-27-C	2908	68	60
18-36-C	3915	64	62
24-36-C	3928	63	65
24-48-C	5220	66	62

In our work, the performance of TT streams is crucial. However, we have not overlooked the BE streams. We strive to ensure their transmission with minimum latency and zero loss. We ensure that the dominance of one stream type does not impact the performance of the other. For example, in scenarios where TT streams are dominant and guaranteed minimal worst-case latency, the BE streams also exhibit acceptable worst-case latency.

To evaluate latencies across varied dominance scenarios and network scales, we illustrate the distribution of total latencies in Fig. 7. As previously highlighted, larger networks with more streams tend to exhibit greater total latency. However, it is observed that the prevalence of one stream type over another has minimal impact on the total latency. This suggests that our model operates effectively across different scenarios. The most marked difference in total latency for networks of similar scale, but varying stream dominance, is seen in the 12-24 scale scenarios. Here, there is a $100\mu s$ discrepancy between the balanced TT-BE and TT-dominant scenarios, representing a 3.94% increase. Notably, this marginal increase doesn't compromise the network's performance.

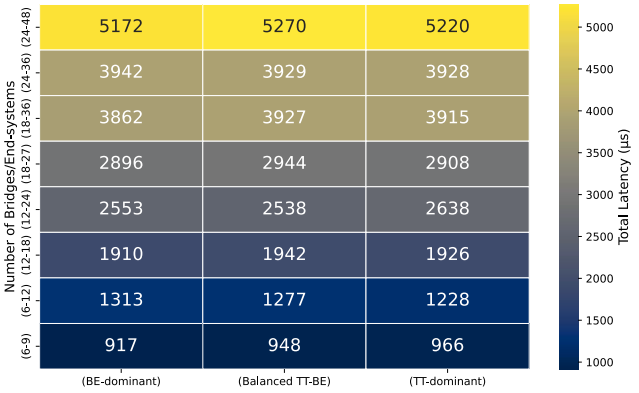


FIGURE 7. Total Latency comparison across three scenarios: BE-dominant, balanced TT-BE, and TT-dominant scenarios, for various scale instances.

2) RESPONSE TIME EVALUATION

Another key metric we evaluate is the response time, which denotes the duration the model solver requires to identify a feasible solution. The total response time encompasses the complete duration needed for the whole procedure, whereas the scheduling and routing response times refer to the durations the scheduling and routing models respectively need for processing. The total response time, scheduling response time, and routing response time for BE-dominant instances are showcased in Table 10; for balanced TT-BE instances in Table 11; and for TT-dominant instances in Table 12.

TABLE 10. The total response time, the scheduling response time, and the routing response time, for the BE-dominant problem instances.

Instance Name	Total Response Time (ms)	Scheduling Response Time (ms)	Routing Response Time (ms)
6-9-A	1284	54.21	1080.79
6-12-A	1897	111.00	1610.06
12-18-A	9987	156.71	8833.36
12-24-A	15963	237.73	14143.91
18-27-A	22934	203.41	19623.62
18-36-A	45607	360.25	41851.81
24-36-A	77201	490.74	71890.85
24-48-A	114431	562.54	104935.49

TABLE 11. The total response time, the scheduling response time, and the routing response time, for the balanced TT-BE problem instances.

Instance Name	Total Response Time (ms)	Scheduling Response Time (ms)	Routing Response Time (ms)
6-9-B	1244	65.81	1014.60
6-12-B	2350	89.76	1999.34
12-18-B	9487	189.56	8172.70
12-24-B	13071	215.40	10999.18
18-27-B	23723	219.06	20947.31
18-36-B	43343	392.67	38690.48
24-36-B	68762	464.48	61823.00
24-48-B	522403	610.14	513068.57

TABLE 12. The total response time, the scheduling response time, and the routing response time, for the TT-dominant problem instances.

Instance Name	Total Response Time (ms)	Scheduling Response Time (ms)	Routing Response Time (ms)
6-9-C	1512	65.56	1245.03
6-12-C	2334	102.22	1997.99
12-18-C	8783	192.07	7832.07
12-24-C	14906	201.18	13484.07
18-27-C	24319	228.96	21725.65
18-36-C	41937	360.68	37209.78
24-36-C	65540	606.14	59632.47
24-48-C	581578	659.20	573310.41

Much like latency, the response time is decisively influenced by the size of the network. The total response time exhibits minor variations between different scenarios of the same scale. For instance, within the 18-36 scale instances, the maximum difference in response time is $3670ms$, observed between the TT-dominant and BE-dominant scenarios. This represents an 8.75% increase for the BE-dominant scenario.

Across all instances and various scenarios, the scheduling response time is significantly lower than the routing response time. The differences in scheduling response times among different scenarios are minimal, though it tends to be higher for the TT-dominant scenarios in larger-scale instances.

The response time exhibited a steady increase up to the 24-48 scale scenarios. For instances within this scale, the BE-dominant scenarios displayed a consistent increase, with the total response time rising by 57.5% compared to the prior 24-36 scale. However, when the count of TT streams surged in the balanced TT-BE and TT-dominant scenarios, there was a pronounced escalation in the total response time—almost 7-times for the balanced TT-BE scenarios and nearly 8-times for the TT-dominant scenarios. Notably, the scheduling model maintained a reasonable response time even at this scale, with the substantial increase primarily attributed to the routing model. This emphasizes the idea that an augmented count of TT streams in larger-scale networks can persuade a steeper response time in the routing model, compared to smaller-scale networks. This influences the total response time. Nevertheless, the model remains efficient, yielding optimal solutions for both the routing and scheduling models.

To thoroughly understand the findings of this study, it is crucial to compare them with previous related research. We will compare our response time results with the previous related work presented in [25] (REU-CP). In that research, the authors tackled the joint scheduling and routing problem using the CP model. They provide an open access to their model enabling us to benchmark our response time findings in opposition to theirs. It is worth noting that they use the term ''optimization time'' instead of ''response time'' in their model. We evaluated all the problem instances from Table 6 using the REU-CP's CP model. The comparisons span various scenarios. For a clearer depiction, we've benchmarked the results across our three distinct scenarios.

In Fig. 8, we compare our total response time results from the first group of scenarios, specifically where streams with BE characteristics more than TT streams. The results indicates that OHDSR outperforms in terms of total response time regardless of the network instance scale. With a reduction of up to 37.21% for the smallest scale instance, 6-9-A, our model demonstrates a notable enhancement in total response time. As we scale to larger instances, our model consistently maintains fine performance, exhibiting up to a 6.63% improvement for the 24-48-A instance.

Similar to the first scenario, when the number of both streams are equal, our model still outperforms in terms of total response time. As depicted in Fig. 9, there is a reduction of up to 12.35%, translating to 9674ms, for the instance 24-46-B—a large-scale network with 24 bridges and 48 end-systems. This reduction is even more pronounced for the smaller scale instances, reaching up to 50.48% for the instance 6-9-B. This trend of enhanced total response time is also observed in instances dominated by TT streams, as illustrated in Fig. 10. For these scenarios, the OHDSR model registers a decrease of up to 26.36% when compared to the REU-CP model.

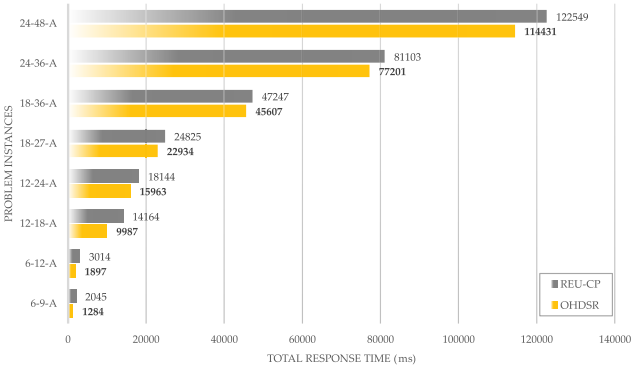


FIGURE 8. The total response time comparison between our proposed OHDSR model and the related work REU-CP [25] model for the BE-dominant problem instances.

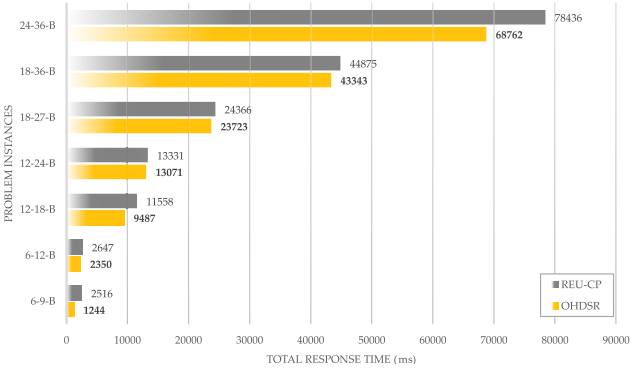


FIGURE 9. The total response time comparison between our proposed OHDSR model and the related work REU-CP [25] model for the balanced TT-BE problem instances.

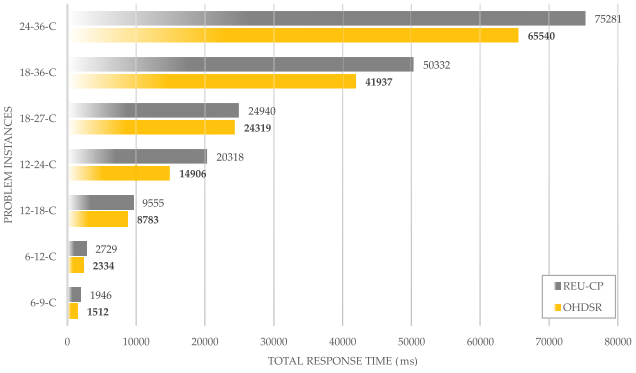


FIGURE 10. The total response time comparison between our proposed OHDSR model and the related work REU-CP [25] model for the TT-dominant problem instances.

For the 24-48-B and 24-48-C instances, we previously observed a significant uptick in total response time compared to the smaller scale instances. Yet, the OHDSR model managed to fully process and provide an optimal solution, maintaining a steady response time for the scheduling model. When running these particular instances on the REU-CP model, the cp-sat solver returned an ''UNKNOWN'' value, indicating no solution was identified. Such consistent results for instances executed on the same system, as shown

TABLE 13. Comparison between our proposed OHDSR model and the related work REU-CP model for the total response time and scheduling response time for the 24-48 scale problem instances.

Instance Name	OHDSR		REU-CP	
	Total Response Time (ms)	Scheduling Response Time (ms)	Total Response Time (ms)	Scheduling Response Time (ms)
24-48-A	114431	562.54	122549	834.92
24-48-B	522403	610.14	/	/
24-48-C	581578	659.20	/	/

in Table 13, indicate that the scalability of OHDSR exceeds that of REU-CP.

Scheduling is a crucial element in achieving the objectives of Time-sensitive networking. Consequently, we aim to compare the performance of our scheduling model with that of REU-CP in terms of response time. This comparison will focus on the same group of scenarios we previously used when evaluating total response time.

In contrast to total response time, the scheduling response time of the OHDSR model exhibits more significant improvement. As illustrated in Fig. 11, our model registers a decrease of 272.38ms for the largest scale instance, representing a 32.62% reduction. For the smallest scale instance, the improvement climbs up to 71.26%. This markedly superior performance is also observed in the second group of instances as shown in Fig. 12, which feature a balanced number of both TT and BE streams. Excluding the 24-48-B instance, which lacks a solution in the REU-CP model, our model demonstrates improvements ranging from 28.21% to 65.13% for this set of instances.

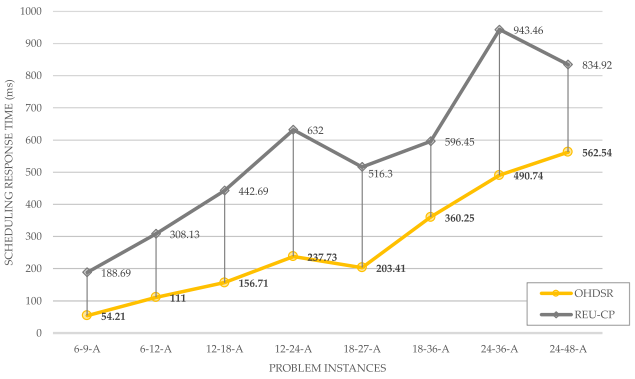


FIGURE 11. The scheduling response time comparison between our proposed OHDSR model and the related work REU-CP [25] model for the BE-dominant problem instances.

Lastly, we examine the scheduling response time for the third group of instances depicted in Fig. 13, which predominantly feature TT streams over BE streams. The OHDSR model maintains its commendable performance when compared with REU-CP, especially for larger scale instances relative to other groups. The 24-48-C instance cannot be compared as it lacks a solution in the REU-CP model.

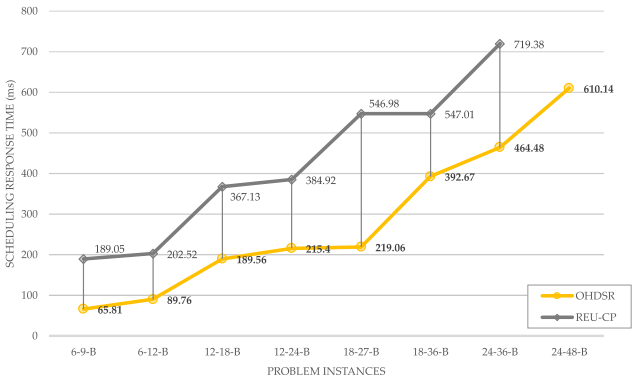


FIGURE 12. The scheduling response time comparison between our proposed OHDSR model and the related work REU-CP [25] model for the BE-dominant problem instances.

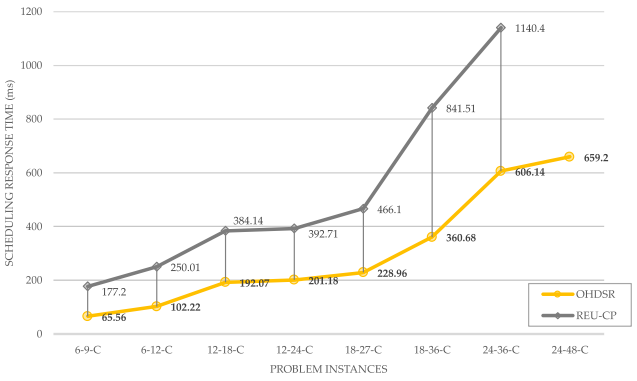


FIGURE 13. The scheduling response time comparison between our proposed OHDSR model and the related work REU-CP [25] model for the BE-dominant problem instances.

The most expansive instance for which we can make a direct scheduling response time comparison between both models is the 24-36-C instance. Here, our model showcases a 46.86% reduction in time, which stands out as a significant decrease for an instance of its size, especially when benchmarked against the 32.62% and 35.44% achieved by the other two instance groups. All these findings highlight the enhanced performance of the OHDSR model in both total response time and scheduling response time compared to the REU-CP model.

VI. CONCLUSION

The evolution of communication networks has heightened the need for real-time communication in numerous sectors, including industrial automation and autonomous vehicles networks. Time-sensitive networking (TSN) emerged as a novel technology, encompassing a set of standards that facilitate deterministic communication over Ethernet networks. This paper addresses the issue of joint scheduling and routing in TSN through the Optimized Hybrid Deterministic Scheduling and Routing (OHDSR) approach. The approach presented is based on constraint programming, which is employed to optimize routing and scheduling by finding the best solution that satisfies the imposed constraints. Along with

the scheduling and routing of Time-Triggered (TT) traffic, Best-Effort (BE) traffic is also considered. The evaluation reveals impressive results concerning the total latency of the traffic and ensures an acceptable worst-case latency for both types of traffic. The experimental evaluation spans various scales of networks with different numbers of bridges and end-systems, and the results highlight the high scalability of our approach. Notably, our model exhibits a markedly improved response time in comparison to similar approaches, with outstanding performance for the scheduling model in terms of response time.

REFERENCES

[1] L. Zhao, P. Pop, Z. Zheng, H. Daigmorte, and M. Boyer, "Latency analysis of multiple classes of AVB traffic in TSN with standard credit behavior using network calculus," *IEEE Trans. Ind. Electron.*, vol. 68, no. 10, pp. 10291–10302, Oct. 2021, doi: 10.1109/TIE.2020.3021638.

[2] 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.

[3] 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.

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

[5] 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.

[6] 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.

[7] *IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks—Amendment 25: Enhancements for Scheduled Traffic*, IEEE Standard 802.1Qbv-2015 (Amendment to IEEE Std 802.1Q-2014 as amended by IEEE Std 802.1Qca-2015, IEEE Std 802.1Qcd-2015, and IEEE Std 802.1Q-2014/Cor 1-2015), 2016, pp. 1–57, doi: 10.1109/IEEESTD.2016.8613095.

[8] *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 (Amendment to IEEE Std 802.1Q-2005), 2010, p. C1-72, doi: 10.1109/IEEESTD.2009.5375704.

[9] 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.

[10] *IEEE 802.1 AV Bridging Task Group*, IEEE Standards 802.1, Institute of Electrical and Electronics Engineers, Piscataway, NJ, USA. Accessed: Jul. 13, 2023. [Online]. Available: https://www.ieee802.org/1/pages/avbridges.html

[11] D. Bezerra, A. T. de Oliveira Filho, I. R. Rodrigues, M. Dantas, G. Barbosa, R. Souza, J. Kelner, and D. Sadok, "A machine learning-based optimization for end-to-end latency in TSN networks," *Comput. Commun.*, vol. 195, pp. 424–440, Nov. 2022, doi: 10.1016/J.COMCOM.2022.09.011.

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

[13] *IEEE Standard for Local and Metropolitan Area Networks—Bridges and Bridged Networks—Amendment 26: Frame Preemption*, IEEE Standard 802.1Qbu-2016 (Amendment to IEEE Std 802.1Q-2014), 2016, pp. 1–52, doi: 10.1109/IEEESTD.2016.7553415.

[14] *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.

[15] 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.

[16] 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.

[17] L. Lo Bello and W. Steiner, "A perspective on IEEE time-sensitive networking for industrial communication and automation systems," *Proc. IEEE*, vol. 107, no. 6, pp. 1094–1120, Jun. 2019, doi: 10.1109/JPROC.2019.2905334.

[18] 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.

[19] 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)*. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers, Aug. 2018, pp. 136–146, doi: 10.1109/RTCSA.2018.00025.

[20] 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.

[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] A. Arestova, K.-S. J. Hielscher, and R. German, "Design of a hybrid genetic algorithm for time-sensitive networking," in *Measurement, Modelling and Evaluation of Computing Systems*, H. Hermanns, Ed. Cham, Switzerland: Springer, 2020, pp. 99–117.

[23] 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.

[24] 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.

[25] 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.

[26] 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.

[27] 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)*. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers, Aug. 2020, pp. 1–11, doi: 10.1109/RTCSA50079.2020.9203662.

[28] 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.

[29] 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)*. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers, Sep. 2018, pp. 337–344, doi: 10.1109/ETFA.2018.8502515.

[30] 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.

[31] V. Gavriluţ and P. Pop, "Traffic-type assignment for TSN-based mixed-criticality cyber-physical systems," *ACM Trans. Cyber-Phys. Syst.*, vol. 4, no. 2, pp. 1–27, Jan. 2020, doi: 10.1145/3371708.

[32] 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)*. Piscataway, NJ, USA: Institute of Electrical and Electronics Engineers, Jun. 2018, pp. 1–4, doi: 10.1109/WFCS.2018.8402374.

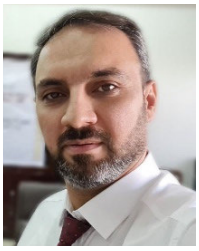
[33] L. Yang, Y. Wei, F. R. Yu, and Z. Han, "Joint routing and scheduling optimization in time-sensitive networks using graph-convolutional-network-based deep reinforcement learning," *IEEE Internet Things J.*, vol. 9, no. 23, pp. 23981–23994, Dec. 2022, doi: 10.1109/JIOT.2022.3188826.
[34] Q. D. Pham 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.
[35] 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.* New York, NY, USA: Association for Computing Machinery, Oct. 2016, pp. 183–192, doi: 10.1145/2997465.2997470.
[36] 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, and J. Millman, Eds., Pasadena, CA, USA, 2008, pp. 11–15.
[37] 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, 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 with the Wireless Sensors Group and his work on wireless medical sensors had gained importance in health care applications involving mobile telemedicine and has had worldwide publicity, including BBC news. He was a Research Consultant with a 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 Ministry of Communications and Multimedia Malaysia and 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 related. Professionally, he has been actively involved as a Lead Auditor of 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 (Student Member, IEEE) received the B.Sc. degree in computer science from the 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, focusing 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.