Design of Time-Sensitive Networks For Safety-Critical Cyber-Physical Systems

Ayman Atallah

A thesis

in

The Department

of

Electrical and Computer Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy (Electrical Engineering) at

Concordia University

Montréal, Québec, Canada

March 2021

Concordia University

School of Graduate Studies

This is to certify that the thesis prepared

By: **Ayman Atallah**

Entitled: **Design of Time-Sensitive Networks For Safety-Critical Cyber-Physical Systems**

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Electrical Engineering)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____ Chair
Dr. Youmin Zhang

_____ External Examiner
Dr. Malek Mouhoub

_____ Examiner
Dr. Hasan Rivaz

_____ Examiner
Dr. Wei-ping Zhu

_____ Examiner
Dr. Rajagopalan Jayakumar

_____ Supervisor
Dr. Otmane Ait Mohamed

Approved by _____
Yousef R. Shayan, Chair
Department of Electrical and Computer Engineering

March 26, 2021 _____
Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

Design of Time-Sensitive Networks For Safety-Critical Cyber-Physical Systems

Ayman Atallah, Ph.D.

Concordia University, 2021

A new era of Cyber-Physical Systems (CPSs) is emerging due to the vast growth in computation and communication technologies. A fault-tolerant and timely communication is the backbone of any CPS to interconnect the distributed controllers to the physical processes. Such reliability and timing requirements become more stringent in safety-critical applications, such as avionics and automotive. Future networks have to meet increasing bandwidth and coverage demands without compromising their reliability and timing. Ethernet technology is efficient in providing a low-cost scalable networking solution. However, the non-deterministic queuing delay and the packet collisions deny low latency communication in Ethernet. In this context, IEEE 802.1 Time Sensitive Network (TSN) standard has been introduced as an extension of the Ethernet technology to realize switched network architecture with real-time capabilities. TSN offers Time-Triggered (TT) traffic deterministic communication. Bounded Worst Case end-to-end Delay (WCD) delivery is yielded by Audio Video Bridging (AVB) traffic. In this thesis, we are interested in the TSN design and verification.

TSN design and verification are challenging tasks, especially for realistic safety-critical applications. The increasing complexity of CPSs widens the gap between

the underlying networks' scale and the design techniques' capabilities. The existing TSN's scheduling techniques, which are limited to small and medium networks, are good examples of such a gap. On the other hand, the TSN has to handle dynamic traffic in some applications, e.g., Fog computing applications. Other challenges are related to satisfying the fault-tolerance constraints of mixed-criticality traffic in resource-efficient manners. Furthermore, in space and avionics applications, the harsh radiation environment implies verifying the TSN's availability under Single Event Upset (SEU)-induced failures. In other words, TSN design has to manage a large variety of constraints regarding the cost, redundancy, and delivery latency where no single design approach fits all applications. Therefore, TSN's efficient employment demands a flexible design framework that offers several design approaches to meet the broad range of timing, reliability, and cost constraints.

This thesis aims to develop a TSN design framework that enables TSN deployment in a broad spectrum of CPSs. The framework introduces a set of methods to address the reliability, timing, and scalability aspects. Topology synthesis, traffic planning, and early-stage modeling and analysis are considered in this framework. The proposed methods work together to meet a large variety of constraints in CPSs. This thesis proposes a scalable heuristic-based method for topology synthesis and ILP formulations for reliability-aware AVB traffic routing to address the fault-tolerance transmission. A novel method for scalable scheduling of TT traffic to attain real-time transmission. To optimize the TSN for dynamic traffic, we propose a new priority assignment technique based on reinforcement learning. Regarding the TSN verification in harsh radiation environments, we introduce formal models to investigate the impact of the SEU-induced switches failures on the TSN availability. The proposed analysis adopts the model checking and statistical model checking techniques to discover and characterize the vulnerable design candidates.

To my parents, wife, and children

Acknowledgments

I cannot thank my supervisor Prof. Otmane Ait Mohamed enough for his support, guidance, and motivation throughout my Ph.D. I also want to thank Dr. Ghaith Bany Hamad for our insightful conversations and for his assistance in my research.

I would like to thank my friends and colleagues at Hardware Verification Group (HVG), specially Abdel-Latif Alshalalfah, Mahmoud Masadeh, and Saif Najmeddin, for creating a kind work environment and making the Ph.D. life more pleasant.

Last but not least, I am deeply grateful for their love, patience, and support to my father, mother, and wife. Special thanks to my brother, Yaman, who has taken huge responsibilities in our family business during my leave for Ph.D.

Contributions of Authors

Article I: *Routing and Scheduling of Time-Triggered Traffic in Time Sensitive Networks*

In this article, I address the following research question: *how to compute no-wait schedules and multipath routings for large-scale Time Sensitive Networks (TSN)?* In this context, I propose three techniques. A scalable iterated scheduling technique based on an Integer Linear Programming (ILP) formulation is proposed. The conflict between messages across iterations is minimized by the proposed partitioning technique, which improves the scheduling success rate. A multipath routing technique is developed to guarantee the fault-tolerance. I am responsible for all the modeling, methodologies applied, experiments, and data interpretation in this article. The co-author in this work (Ghaith Bany Hamad) contributed in the writing of the article and reviewing the early draft of the manuscript.

Article II: *Dynamic TSNs Priority Assignment for Fog Computing Using Reinforcement Learning*

In this article, I propose a dynamic flow priority assignments approach for TSN to meet the requirements of dynamic systems, such as Fog computing applications. I formalize the priority assignment problem as a reinforcement learning problem. An agent is trained to meet the timing constraints while maximizing the overall network

throughput. This problem is addressed by the double deep Q-network technique to find the optimal priority assignment policy. I am responsible for all the modeling, methodologies applied, experiments, and data interpretation in this article. The co-author in this work (Ghaith Bany Hamad) contributed in the writing of the article and reviewing the early draft of the manuscript.

Contents

List of Figures

List of Tables

List of Acronyms

AVB Audio Video Bridging. iii, 2, 3, 5, 10–12, 15–17, 19, 24, 29, 46, 77

BE Best Effort. 3, 11, 46, 80

CA Configuration Agent. 75

CBS Credit-Based Shaper. 3, 12, 81

CEV Crew Exploration Vehicle. 26, 100, 104

CGC Cross-Group Conflict. xiv, 55, 56, 59, 66

CNC Central Network Configurator. 79, 82, 85, 87

CPS Cyber-Physical System. iii, iv, 1, 2, 4–8, 10, 13, 16, 24, 45, 49, 74, 110, 111

DA/IRS DoC-Aware Iterative Routing and Scheduling. xiv, xv, 47, 48, 65, 66, 68–72

DAMR DoC-Aware Multipath Routing. 47, 48, 57–59, 66, 72

DASP DoC-Aware Stream Partitioning. xiv, 47, 48, 54, 55, 57–59, 64–67

DoC Degree of Conflict. 47, 53, 54

DRS Disjoint Routing Set. 57–59

DTI Diagnostic Test Interval. 28, 30

DTRR Detached Temporal Redundant Routing. 29

EN Edge Node. 78

ES End System. xiii, xiv, 4, 11, 19, 33–36, 40, 41, 46, 49, 51, 65, 98, 100, 104

FIFO First-In First-Out. 11, 79, 80

FPGA Field-Programmable Gate Arrays. 100

FRER Frame Replication and Elimination for Reliability. 46, 49

GCL Gate Control List. 11, 17, 78

IIS Iterated ILP-based Scheduling. 47, 48, 59, 61, 64, 66, 72

ILP Integer Linear Programming. 47, 76

ILP/JRS ILP-based Joint Routing and Scheduling. xiv, xv, 68–72

ILPMPR ILP-based Multiple-Path Routing. 24, 37, 43

ILPTRR ILP-based Temporal Redundant Routing. 29

MDP Markov Decision Process. 87

ML Machine Learning. 78

MT Multipath Transmission. 5, 15–17

MTTDE Mean Time To Detect Error. 30, 31, 40, 41

MTTF Mean-Time-To-Failure. 15–17, 28, 49

MTU Maximum Transmission Unit. 51

nCGC normalized CGC. 56–58

NLR Network Level Redundancy. 3, 5, 34

nTC not Time Critical. 79

PA Priority Assignment. 12, 75–78, 81, 82, 85, 87, 92, 96

PB/JRS Pseudo-Boolean Joint Routing and Scheduling. xiv, xv, 68–72

PCP Priority Code Point. 11

RATS Redundancy Aware Topology Synthesis. 19, 20, 33

RBD Reliability Block Diagram. 16

RC Rate Constrained. 49, 77, 78, 80, 82, 111

RL Reinforcement Learning. 8, 18, 77, 78, 88

RL/DPA Reinforcement Learning-based Dynamic Priority Assignment. xv, 76, 84, 85, 92, 94–96

RSP Random Streams Partitioning. xiv, 47, 48, 53, 54, 65–67

RTS Reserved Time-Slots. 62

SE Sensitive Entry. 105

SEU Single Event Upset. iv, 7, 8, 111

SMC Statistical Model Checking. 101

SMT Satisfiability Modulo Theories. 47, 48, 77

SW Switch. 78

TA Timed Automata. 101

TAS Time Aware Shaper. 2, 11, 111

TC Time-Critical. 79, 81

TMRS Truncated Multipath Routing Space. 58

TSN Time Sensitive Network. iii, iv, 2–4, 6–8, 11, 13, 16, 17, 19, 24, 45, 47–49, 72, 75, 78, 80, 82, 110

TSSDN Time-Sensitive Software-Defined Networks. 48

TT Time-Triggered. iii, iv, 2–5, 7, 8, 10, 11, 15, 16, 46, 55, 72, 77, 80, 111

UBS Urgent-Based Scheduler. 15, 77

VID VLAN Identifier. 11, 12

VM Vulnerable Message. 105

WCD Worst Case end-to-end Delay. iii, 2, 12, 46, 82, 102

Chapter 1

Introduction

Recent years have seen a massive increase in the computation and communication capabilities offered by modern technologies. Such technologies emerge the transition toward CPS, which integrates digital computations control and monitors physical processes in a distributed architecture (Poovendran, 2010). CPSs include safety-critical applications in which system failures can lead to interruption of services, financial losses, and even loss of life. CPSs are increasingly deployed in many safety-critical areas such as automotive and avionics to replace mechanical systems (Lee, 2008). The scope of CPS for safety-critical applications covers critical infrastructures such as power grid and highway transportation system, energy, and industrial automation system. Moreover, healthcare and bio-medical system are also examples on safety-critical CPSs (Poovendran, 2010). In contrast to digital systems, CPSs not only depend on the correctness of computed variables, but also on the physical time when their values are delivered (Kopetz, 2011). For example, the correctness of transferring a file by email is not affected by a 1-second delay, while the correctness of a control message belongs to braking systems can be affected by 10 ms of an unexpected delay. Therefore, time-sensitive communication must guarantee deterministic message

delivery or very low latency for CPSs in safety-critical applications.

1.1 Time-Sensitive Networking

Bus-based and switched networks can deliver low latency communication. Each solution has different pros and cons. The bus-based networks need a more straightforward design than switched networks, but on a limited scale. For example, the available bus standards, such as CAN and FlexRay bus offer a bandwidth < 20 Mbps (Etschberger, 2001; Makowitz & Temple, 2006). Yet, such bandwidth meets the CPSs demands in many applications. Nevertheless, future applications would require higher bandwidth and broader coverage areas, which would not be achievable by bus-based networks. On the other hand, the switched architecture, such as Ethernet, provide large bandwidth in the range of 1 Gbps. Hence, switched networks become essential to match the increasing complexity of future CPSs (Tămaş-Selicean, Pop, & Steiner, 2015). However, adapting switched architecture for time-critical communication is challenging. For instance, the non-deterministic queuing delay over multiple hops and the packet collisions deny low latency communication in conventional Ethernet.

TSN standard has been introduced to get rid of the timing limitations of switched networks which meets the requirements of CPSs (IEEE, 2014). Adapts the Ethernet technology allows TSN to provide high bandwidth with low cost. TSN supports two types of transmission for critical traffic, namely, TT or AVB transmission. TT transmission provides deterministic delivery with very low latency, (e.g., $< 100\mu$s). Whereas, AVB transmission guarantees bounded low WCD delivery, (e.g., < 2ms). TT messages, which are also known as scheduled messages, are transmitted according to a fixed schedule that guarantees exclusive access to the transmission queue at particular time slots. Each egress port in the network deploys a Time Aware Shaper

(TAS) as shown in 1.1 to block the non-scheduled traffic to reach switches output ports during those time slots. On the other hand, AVB messages are transmitted according to a Credit-Based Shaper (CBS) that shapes the transmission rate and prevents bursts and starvation of the lower priority AVB and Best Effort (BE) traffic. AVB type is an asynchronous traffic which implies less complexity than TT traffic. The messages of non-critical applications are generally classified as BE traffic which has the lowest priority and does not guarantee timing characteristic.

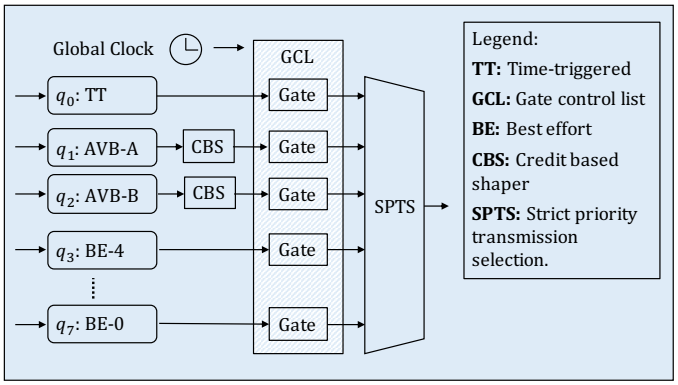


Figure 1.1: Egress port with IEEE 802.1Qbv enhancements.

The safety-critical applications require an instantaneous fault-tolerance communication to ensure proper operations (Kehrer, Kleineberg, & Heffernan, 2014). Thus, the conventional on-demand fault-tolerance techniques, such as packets, re-transmission, and network reconfiguration, cannot be employed. Instead, TSN attains the transmission reliability by a seamless redundancy approach, i.e., sending multiple replicas of the critical messages simultaneously. The seamless redundancy can be achieved by deploying complete redundant networks, i.e., Network Level Redundancy (NLR). The NLR approach is simple but inefficient in terms of power, weight, and cost.

1.2 Problem Statement

This section explains the challenges of TSN design that impede TSN's deployment in realistic safety-critical CPSs. Such systems impose a large variety of constraints, i.e., no single design approach fits all applications. For example, some applications need strict reliability constraints for safe operations. These constraints require a multipath routing where redundant packets are continuously transmitted over disjoint paths. However, it is not practical to adopt such a cost-intensive approach for less critical applications. Therefore, TSN's efficient and practical employment demands a flexible design framework that offers several design approaches to meet the broad range of timing, reliability, and cost constraints of CPSs.

The first challenge is related to the design space exploration. The capacity gap between the existing design techniques and the size of new systems is widening. The upcoming CPSs tend to accommodate a vast number of ESs that exchange hundreds or even thousands of time-critical messages. Consequently, the underlying networks' size would be substantial in terms of switches, connections, and bandwidth. However, scalability is a common limitation in all existing techniques. The scheduling of TT messages is the bottleneck in terms of the scalability since it is NP-hard problems (B. Wang & Hou, 2000). In particular, the runtime hikes by increasing the network size, even for small test cases, e.g., 50 messages over a 4-switches network. For instance, the networks reportedly tackled by the existing TSN design techniques consist of less than 20 ESs and 100 messages (see Section 2). On the other hand, some advanced vehicles nowadays deploy more than 50 ESs, exchanging hundreds of critical messages (Chakraborty et al., 2012).

The second challenge is to guarantee the reliability of TSN networks. Addressing the reliability aspect at the early stage of the design allows us to satisfy it in a

resource-efficient manner. For example, late handling of the reliability may lead to critical situations where the remaining network bandwidth is not enough for reliability mechanisms or timing requirements are too tight to add those mechanisms. Therefore, it is vital to develop a systematic approach to address reliability at early design stages and with all other design constraints. Otherwise, resource-intensive approaches, such as NLR, would be necessary. The step from non-redundant routing to Multipath Transmission (MT) routing comes with an enormous design space growth. All existing work either assume the reliability is attained by NLR approach or they neglect TT traffic and only adapt MT for AVB traffic.

The third challenge is how to employ the TSN for dynamic traffic? As we mentioned before, TSN is originally introduced for fixed traffic with static network configurations. Static networks meet the requirements of a wide range of applications, such as automotive applications. On the other hand, an emerging number of real-time applications imply dynamic traffic, such as fog computing and storage (Pop, Raagaard, Gutierrez, & Steiner, 2018). Such applications require a dynamic TSN reconfiguration in which networks accommodate and remove packet streams on the runtime (Haur & Chin, 2019). An efficient implementation of fog computing requires efficient utilization of the network assets. In other words, accommodating more flows allows a more significant portion of tasks to be executed by fog nodes. Thus, dynamic traffic needs reconfiguration algorithms that meet the hard deadlines of the time-critical traffic while maximizing the network's long-term throughput. Nonetheless, such algorithms have been left out of the TSN standard.

The fourth challenge is related to the uncontrolled environments in which CPSs are deployed in the field. These environments restrain abnormal conditions and post challenges to verify the running safety-critical CPSs. One example of such conditions

is the harsh radiation environment, which affects CPSs in space and avionics applications. The high energetic particles result in SEU-induced failures. Failure due to bit flips in memory is well knows phenomena that have affected projects such as NASA's Cassini Spacecraft (Morgan, 2016) and SpaceX's Falcon 9 (Leppinen, Kestilä, Tikka, & Praks, 2016). Also, systems closer to earth are vulnerable to bitflips (Normand, 1996). Such failures result in unexpected timing behavior in the network, which can jeopardize the running CPSs (Kauer, Soudbakhsh, Goswami, Chakraborty, & Annaswamy, 2014). Neither the TSN reliability nor the control stability under faulty switches has been addressed in the literature.

1.3 Thesis Contributions

This thesis aims to push forward the on-going research efforts on enabling TSN deployment in a broad spectrum of CPSs. We achieve this goal by developing a TSN design framework that addresses the reliability, timing, and scalability aspects. Topology synthesis, traffic planning, as well as early-stage modeling and analysis are considered in this framework. The proposed methods work together to meet a large variety of constraints in CPSs. However, each method does its role independently and can be integrated with other related works. The main contributions of this work are listed as follows:

- Development of a scalable heuristic-based method for fault-tolerance topology synthesis. The proposed method ensures several disjoint paths between sources and destinations while considering the timing requirements of TT traffic. The redundancy level can be adjusted based on the criticality of the traffic. The method is explained in Chapter 3 and has been published in [Cf4].

- Introducing ILP formulations for reliability-aware AVB traffic routing. The proposed methods adapt either spatial or temporal redundancy to meet the reliability constraints for different CPSs. The formulation and the experimental results are explained in Chapter 3 and have been published in [Cf1] and [Cf5].

- Development of a novel method for a scalable scheduling of TT traffic. The proposed method can handle large networks to meet the requirements of the upcoming realistic CPSs. An iterated ILP-based scheduling is adopted for scalability while the degree of conflict between iteration is minimized using a graph-based technique. This work is explained in Chapter 4, and has been published in [Jr1].

- Employing a machine learning technique, namely reinforcement learning to optimize the TSN design. The proposed method enhances the capability of TSN to accommodate dynamic traffic which can be generated in Fog computing and storage application. This method is explained in Chapter 5 [Jr2].

- Introduce formal models to investigate the impact of the SEU-induced faults on the TSN. The proposed methods analyze the network reliability and availability under a harsh radiation environment. Such an analysis allows dismissing vulnerable designs at early stages. The proposed analysis methods are explained in Chapter 6 and have been published in [Cf2] and [Cf3].

1.4 Thesis Outline

The rest of the thesis is organized as follows:

- **Chapter 2 - Background and Prior Work:** This chapter is twofold, containing a brief background on the TSN, and the required prior work relevant to

the design challenges addressed in this thesis.

- **Chapter 3 - Reliability-Aware TSN Design:** This chapter explains the fault-tolerant approaches proposed in this thesis to meet a range of various CPSs requirements.

- **Chapter 4 - Article I - Routing and Scheduling of Time-Triggered Traffic in Time Sensitive Networks:** This chapter demonstrates the scheduling challenge and introduces the proposed no-wait scheduling technique for TT traffic in TSN.

- **Chapter 5 - Article II - Dynamic TSNs Priority Assignment for Fog Computing Using Reinforcement Learning** This chapter explains two TSN optimization techniques based on Reinforcement Learning (RL) approach to enhances the TSN design scalability, and optimize the TSN performance for Fog applications.

- **Chapter 6 - TSN Verification Under Single Event Upsets:** This chapter investigates the impact of harsh radiation environment on TSN. Formal modeling of TSN is introduced to enable reliability and availability analysis under several SEU-induced failure scenarios.

- **Chapter 7 - Conclusion:** This chapter concludes the thesis and discusses future research directions.

Chapter 2

Background and Prior Work

This chapter introduces the fundamentals that are required for understanding the thesis contributions. Moreover, we introduce the related work in real-time communication, dynamic configuration, and network verification to demonstrate our work context.

2.1 Real-Time Communication

Cyber-physical systems for safety-critical applications increasingly rely on a distributed architecture for scalability. Closed-loop control systems need real-time data streamed from distributed sensors to monitor the physical environment. The collected measures are exchanged between distributed computing nodes to computing the proper response. Then, the computed control signals are delivered periodically to the actuators to maintain system stability. The control cycle period can be very short, i.e., in the range of milliseconds in safety-critical applications, such as automotive, avionics, and industrial automation. The proper operation of such systems is susceptible to the reliability and timing of the underlying communication network. Thus,

the network must provide a deterministic message delivery or at least a predictable timing with a bounded end-to-end delay. Bus and switched architectures are viable solutions to yield such real-time communication with different features. The former option implies simpler design processes, while the latter one offers higher scalability. The bus architecture is introduced in several standards, e.g., CAN and FlexRay. (Etschberger, 2001; Makowitz & Temple, 2006). Yet, bus networks meet the communication demands of some CPS in terms of bandwidth and coverage. An increasing number of complex systems require large bandwidth and broader coverage beyond the bus networks' capabilities. Hence, switched networks become essential to support the rising demand for modern CPSs (Tămaş-Selicean et al., 2015). Ethernet is an attractive option to implement switched networks due to large bandwidth, high scalability, and efficient cost (Tămaş-Selicean et al., 2015). However, standard Ethernet does not provide the timing guarantees required by real-time applications (Decotignie, 2005). The IEEE Time-Sensitive Networking Task Group has introduced IEEE 802.1 standard to remove such limitations Ethernet networks (IEEE, 2014). TSN allows mixed-critical traffic by supporting two classes of time-critical traffic, namely, TT or AVB, in addition to the non-time-critical class. This feature allows resource sharing, which results in a cost-efficient networking solution.

2.1.1 Time-Triggered Traffic

TSN employs a communication schedule to attain deterministic message delivery. The communication schedule specifies the time at which messages are to be sent and forwarded along its route. Such transmission scheme requires a precise global clock. The messages that require such synchronization between nodes are known as time-triggered messages. IEEE 802.1 standard defines new mechanisms and hardware to

enable TT communication over Ethernet networks. The TSN is interconnected by full-duplex physical links, which allow simultaneous transmission in both directions and eliminates collisions. The messages are exchanged between ESs in TSN under the concepts of streams and frames. The frame represents any message instance, while the stream represents a flow of frames transmitted from a particular source to one or more destinations with a specific path, frequency, size, and priority. Every frame header has a VLAN Identifier (VID) of 12 bits indicate the stream to which it belongs, in addition to 3 bits that indicate the corresponding stream's priority, namely Priority Code Point (PCP). TSN follows a strict-priority preemptive forwarding scheme.

The deterministic delivery of TT traffic is guaranteed by two means; global overlap-free schedule and precise clock synchronization as defined in IEEE 802.1AS or IEEE 1588. These standards allow sub-microsecond synchronization of clocks in measurement and control systems. Several frames of each message can be transmitted during the hyper-period, which is the least common multiple of all messages' periods. The schedule defines the time offset at which each frame is transmitted at the hyper-period. Furthermore, the schedule defines the instant time at which the TT frame is forwarded through switches along its path. The schedule is realized by the TASs, which are deployed on each output port. A simplified schematic of TAS is shown in Fig. 1.1. The TAS adopts a simple First-In First-Out (FIFO) queuing paradigm and isolates the traffic classes in several separate buffers according to the PCP value in the frames' headers. Each buffer ends by a timed gate. The opening and closing sequences of each gate are stored in the Gate Control List (GCL). The TAS closes the AVB and BE gates before the arrival of the scheduled TT frames to eliminate the queuing delay.

2.1.2 Audio-Video Bridging Traffic

TSN introduces the asynchronous AVB traffic class which guarantees bounded WCD and zero congestion loss. Such reliability and timing guarantees are achieved using the CBS, which regulates the traffic flow to a predefined burstiness and rate. The AVB class meets the requirements for various time-sensitive applications coexisting with non-time-critical best-effort traffic. Instead of configuring the CBS for individual streams, a particular priority level is assigned to each stream using the frames' VID tags. The CBS regulate the frame forwarding according to their priority. The asynchronous paradigm implies more straightforward implementation, i.e., frames have neither periodic patterns nor specific offsets. Such a paradigm eliminates the complexity of precise timing synchronization based on a global clock. However, a flow must adhere to predefined transmission rate and burstiness. The source adheres to a contract that regulates the frame emission rate of each stream. The CBS reduces AVB traffic latency by blocking the sources that exceed the reserved bandwidth assigned by the contract. Such that, the cumulative data $w_i(d)$ of flow f_i with a burstiness $\hat{b}_i$ over time interval d is limited to:

$$w_i(d) \leq \hat{b}_i + (\hat{r}_i \times d) \tag{1}$$

where $\hat{r}_i$ is the designated leaky bucket rate for flow f_i. The CBS apply a packet-by-packet leaky bucket algorithm (Goyal & Vin, 1997; Zhang & Ferrari, 1993) to regulate the flows' transmission rate over the hops along their paths as shown in Fig. 2.1. Fig. 2.1 shows two different patterns (A & B) of flows that share the same parameter, i.e., burstiness and rate. Each flow f_i is assigned to a certain priority $p_i \in [0, K)$ which applies on all servers over its route where $\mathcal{P} = \{p_i | f_i \in \mathcal{F}\}$ denotes the flow Priority Assignment (PA) in the whole network.

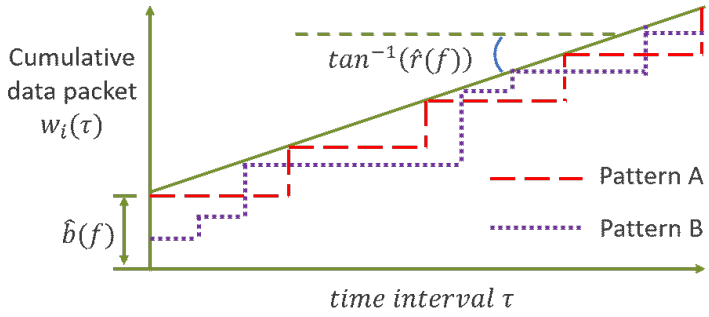


Figure 2.1: An example of two traffic patterns that satisfy the same leaky bucket constraint.

2.2 Scheduling Problem

CPSs typically have a static architecture that results in regular traffic through the TSN. Thus, unlike the other communication networks, the topology synthesis and the traffic routing and scheduling of TSN networks can be planned at design time. The scheduling problem for TT traffic is tackled in (Craciunas, Oliver, Chmelík, & Steiner, 2016), which presents an SMT approach to support the IEEE 802.1Q amendment for scheduled traffic. The authors formally define the required constraints for valid deterministic schedules that provide jitters-free transmission and deterministic end-to-end latency guarantees for strictly-periodic scheduled frames. The technique above handles the routing and the scheduling tasks separately despite the interrelation between these two tasks, which disregards possible schedules and limits the optimization capabilities. Several works have addressed the joint routing and scheduling synthesis. Authors in (Smirnov, Glaß, Reimann, & Teich, 2017b) introduce a *pseudo boolean* constraints formulation for the joint generation of both valid routing and schedule in a single step. The authors demonstrate the proposed formulation's capabilities by performing multi-objective routing and scheduling optimization based on a genetic algorithm. In the same context, (Schweissguth, Danielis, Timmermann, Parzyjegla,

& Mühl, 2017) introduces an ILP formulation that solves the routing and scheduling problem for TT traffic jointly. In addition to generating a valid (overlap-free) schedule, This formulation satisfies the messages' deadlines and minimizes the overall sum of messages' latency. Several assumptions are used to simplify this problem, such as assuming that the links are identical, i.e., each message has the same transmission time and link delay on all links. The authors show that the proposed approach outperforms the 2-steps approach in terms of providing lower latency.

In order to enhance scalability, several heuristic solutions are investigated. Authors in (Pop, Raagaard, Craciunas, & Steiner, 2016) introduce a design optimization framework for both TT and AVB traffic in the TSN network, such that an overlap-free schedule is synthesized and the WCD of AVB traffic is minimized. The routing is generated by a greedy randomized adaptive search procedure-based heuristic. Then, the TT messages schedule is determined using an ILP-based algorithm, limiting this approach's scalability. An SMT-based heuristic approach for joint routing and scheduling for TT traffic in TSN networks is introduced in (Mahfuzi et al., 2018a). The proposed approach generates stability-aware solutions such that all TT messages match the maximum latency that guarantees their control applications to be stable. The authors adopt two heuristics to solve the scalability problem of SMT formulation. First, k-shortest paths for each message are considered for the routing instead of handling the whole set of the possible paths, which may be huge in densely connected large-scale networks. Second, the global solution is built incrementally by dividing the set of TT messages into several sub-sets, i.e., the SMT solver handles one sub-set at a time. All of the techniques mentioned above suffer from scalability limitations.

2.3 Routing and Topology Synthesis

The multipath transmission approach has been studied in (Smirnov, Reimann, Teich, Han, & Glaß, 2018) which introduces two techniques for multipath routing optimization. An SAT-based formulation for the routing problem is introduced. For the sake of further simplification, authors assume that the network topology is given in advance. Authors in (Kehrer et al., 2014) provide an excellent overview of the fault-resilience approaches for TSN. In (Jeon, Lee, & Park, 2015), authors propose a dual-path method to enhance the timing performance of critical traffic in TSN. The method exploits the redundant paths to send different portions of the original stream instead of sending identical copies of the whole stream. The authors present several scenarios where the proposed technique enhanced the performance. A mathematical analysis of the gain in the transmission reliability due to temporal redundancy is introduced in (Smirnov, Glaß, Reimann, & Teich, 2016). The techniques mentioned above do not guarantee multiple disjoint paths between nodes exchange critical messages. Therefore, the introduced approach is limited to statistically optimize transmission reliability in terms of Mean-Time-To-Failure (MTTF). To address load-balancing and congestion in TSN, (Ojewale & Yomsi, 2020) propose two routing heuristics that aim to find a feasible route for each flow, i.e., the traffic on each link is minimized.

To guarantee the redundancy requirements of critical traffic, authors in (Gavrilut, Zarrin, Pop, & Samii, 2017) propose a fault-tolerant topology synthesis approach. Heuristic and constrained programming-based techniques for topology planning and traffic routing are introduced. Both techniques support a particular class of AVB traffic, namely, Urgent-Based Scheduler (UBS) traffic, which does not provide deterministic delivery because it is not transmitted according to overlap-free scheduling tables. Most of the literature neglects the MT approach for TT traffic. Neglecting

the TT traffic can be due to the vast complexity implied by the scheduling task. Nonetheless, TT traffic has the best timing behavior in TSN network which is desired in the safety-critical CPSs. Hence, a new technique support the MT fault-tolerance approach for TT traffic is essential.

2.4 Networks Verification

A simulation-based verification framework is introduced in (Heise, Geyer, & Obermaisser, 2016) based on OMNeT++ software. The framework evaluates the main features presented in TSN, including multipath transmission. Other features, such as frame preemption as well as per-stream policing, are presented as well. The interrelation between reliability and timing constraints is investigated. The impact of multipath transmission and the frame preemption on the timing behavior of both TT and AVB traffic is studied. The simulation-based analysis can tackle detailed models while providing a good impression of the average case within a relatively short runtime. However, it is incapable of performing exhaustive exploration for the whole search space. In other words, simulation-based approaches are not adequate to guarantee the desired reliability and timing properties of safety-critical systems.

In the context of Ethernet-based switched networks, authors in (Kanabar & Sidhu, 2009) investigate the reliability of several network topologies such as ring and star-ring. An analytic approach is used to compute the network reliability based on the Reliability Block Diagram (RBD) model. Failures in switches, as well as the time synchronization clock, are considered. The failure scenarios in the switches and global clock are abstracted as MTTF of the component. The conducted level of abstraction simplifies the analysis and allows the handling of large networks in a short time. However, this approach is incapable of representing the impact of a particular failure

in a specific switch on the transmission reliability of other paths in the network. The reliability analysis considering the specification of avionics applications is addressed in (C. Wang, Li, & Hu, 2011). The system is modeled as a fault tree under different failure scenarios in switches, end-systems, and links. The fault tree model is analyzed according to the failure distribution information of each essential event. A mathematical analysis of the transmission reliability under temporal redundancy in TSN network is introduced in (Smirnov et al., 2016). This work illustrates the trade-off between the temporal redundancy and network congestion in terms of MTTF of AVB messages. Nevertheless, none of the literature addresses the transmission reliability of TSN networks under MT fault-tolerance approach.

2.5 Dynamic Reconfiguration

Pieces of literature have been studied the dynamic configuration topic, i.e., enhancing the reliability or meeting the dynamic traffic requirements. A dynamic configuration agent for Fog applications optimization is introduced in (Gutiérrez, Ademaj, Steiner, Dobrin, & Punnekkat, 2017; Gutiérrez, Steiner, Dobrin, & Punnekkat, 2015). The proposed agent allocates new TT flows generated by Fog computing systems on runtime. Such agents observe the transmission patterns of new flows to estimate their period, length, and latency, enabling them to find an optimized schedule. The runtime scheduling of TT traffic in TSN is addressed in (Raagaard, Pop, Gutiérrez, & Steiner, 2017) by a greedy heuristic technique for fast execution. This technique determines the GCL, which decides the transmission of TT frames on each egress port of a network switch. The authors in (Mai, Navet, & Migge, 2019; Navet, Mai, & Migge, 2019) investigate the speed up of the design space exploration in TSN using machine learning in replacement of the conventional schedulability analysis, e.g.,

network calculus. Supervised and unsupervised machine learning techniques are employed to test the schedule feasibility for a particular offline configuration. The RL is employed in (Y. Wang, Wang, Huang, Miyazaki, & Guo, 2018) for dynamic traffic and computation co-Offloading in Fog computing implemented for mobile services in vehicular networks. The trade-off between energy consumption and service delay is investigated by dynamic RL-based and deep RL-based scheduling techniques.

Regarding the fault-tolerant aspect, (Desai & Punnekkat, 2020) aim to enhance the bandwidth utilization by avoiding the message replications under good link conditions. Instead, they investigate the frame's duplication only when a link has a higher propensity for failure. Since the TSN standard does not define fault detection mechanisms, the authors propose a machine learning-based intelligent configuration synthesis mechanism to estimate the links' propensity for failures. In (Pahlevan, Schmeck, & Obermaisser, 2019), the authors propose a fully centralized model for the dynamic configuration of safety-critical systems. A central agent monitors the network and detects topology changes. If a faulty link or node is detected, the agent remotely re-configures the TSN switches via the spanning tree protocol messages. Such dynamic reconfiguration offers a lightweight fault-tolerance for critical messages under specific failure scenarios. However, the introduced method uses best-effort messages to monitor and control the network, which cannot attain a real-time recovery. Such a limitation denies this method to be deployed in many safety-critical applications.

Chapter 3

Reliability-Aware TSN Design

In this chapter, we propose three techniques for topology and routing synthesis. The proposed techniques aim to meet the redundancy requirements of critical messages in TSN efficiently. First technique performs topology synthesis considering the spatial redundancy constraints in TSN (Section 3.1). Second technique, generates multipath routing for AVB messages that require such a redundancy (Section 3.2). Third technique investigates the temporal redundancy approach for increasing overall TSN reliability (Section 3.3).

3.1 Fault-Tolerant TSN Topology

To meet the required spatial redundancy of critical messages, we introduced the Redundancy Aware Topology Synthesis (RATS) technique. The inputs of RATS are the ESs set and the traffic between them. The traffic specifications include the required redundancy for each message. The technique synthesizes an optimized topology that offers multiple disjoint paths between ESs. The number of the disjoint paths between nodes depends on the required messages' redundancy. In addition to

the redundancy constraints, RATS decides the required bandwidth of each link in the synthesized network. The appropriate network components will be selected to satisfy the constraints mentioned above and optimize the total cost. The cost is defined by a library that contains the available components, i.e., switches, and links. The component library defines the number of ports, the bandwidth, and the cost of each switch and link. Such a cost can be monetary cost, power consumption, or any other criteria. Example libraries are shown in Table 3.1 and Table 3.2, for switches and links, respectively.

3.1.1 Proposed Topology Synthesis Algorithm

The synthesis problem is formulated as a labeling problem based on the graph model. Given the set $\mathcal{B} = \{b_1, b_2, b_3, ..., b_K\}$ and the set $\mathcal{L} = \{l_1, l_2, l_3, ..., l_N\}$ that represent the bridges in the network topology and the links which connect these bridges, respectively. K is an upper bound for the number of bridges specified by the user and N is the number of links in a fully-connected network which equal to $K(K-1)/2$. Hence, the set $\mathcal{D} \in \mathcal{B} \cup \mathcal{L}$ represents the decision elements that can be parts of the network topology. Then, let the vector $\mathcal{T} = \{t_{d_1}, t_{d_2}, t_{d_3}, ...t_{d_{(K+N)}}\}$ be a labeling vector whose element t_{d_i} represents the assignment of a label to the element $d_i \in \mathcal{D}$. Each element $t_{d_i} \in \mathcal{T}$ specifies the type of d_i from the library $\mathcal{A}$, whereas $t_{d_i} = \emptyset$ means that t_{d_i} is discarded element. Hence, the vector $\mathcal{T}$ describes the synthesized network topology.

The main steps of RATS are illustrated in Algorithm 1. In each iteration, the algorithm can modify the topology by adding new components if needed to route the new message. The first step in the algorithm is to determine an ordered list of messages $\mathcal{M}_\psi$ using the function *orderMessages* in line 3. Two criteria are followed

Module ID	Number of Ports	Cost
b_1	2	3
b_2	3	6
b_3	4	8

Table 3.1: Example Library for Bridges Modules

Link ID	Bandwidth (Mbps)	Cost
l_1	100	2
l_2	1000	5

Table 3.2: Example Library for Physical Links

to order $\mathcal{M}$: i) frame frequency where the messages with more frequent frames are assigned first since the messages with lower frequency are easier to schedule; and ii) frame size where the frames of the same frequency, larger frames are assigned first since the smaller ones are easier to schedule.

Algorithm 1: Redundancy Aware Topology Synthesis

Input : $\{\mathcal{E}, \mathcal{M}, \mathcal{A}, N_{\mathcal{B}}\}$
Output: $\{\mathcal{T}, \mathcal{R}, \mathcal{S}\}$

$\mathcal{W} \leftarrow \mathcal{W}_{init}$
$\mathcal{R} \leftarrow \emptyset$, $\mathcal{S} \leftarrow \emptyset$
$\mathcal{M}_\psi \leftarrow oderMessages\ (\mathcal{M})$
while $\mathcal{M}_\psi \neq \emptyset$ **do**
  $\mathcal{W}_m \leftarrow removeJointPaths(\mathcal{W}, R_m)$
  $flag \leftarrow 0$, $k \leftarrow 0$
  while $flag == 0$ **do**
    $k \leftarrow k + 1$
    $r_m^k \leftarrow findNextPath\big(G(\mathcal{V}, \mathcal{W}_m), m, k\big)$
    $[\mathcal{S}, flag] \leftarrow assignMessage(\mathcal{S}, m, r_m^k)$
  end
  $R_m \leftarrow R_m \cup r_m$
  $L^m_{tolerance} \leftarrow L^m_{tolerance} + 1$
  if $L^m_{tolerance} == m.tl$ **then**
    remove m from $\mathcal{M}_\psi$
  end
  $\{\mathcal{W}, \mathcal{T}\} \leftarrow updateWeights(\mathcal{W}, \mathcal{T}, r_m)$
end

For every message in $\mathcal{M}_\psi$, one path would be added, i.e., it is disjoint with the paths added in previous iterations. Given that R_m is the set of the selected paths

for message m, the function *removeJointPaths* generates a message-specific arcs set $\mathcal{W}_m$. In this set, the links belong to R_m are disabled by having *infinite* weights. This ensures that the new generated path is disjoint with all links in R_m. Then, the function *findNextPath*, which based on Yen's algorithm (Yen, 1970), finds r_m^k which is the k^{th} shortest path for message m through $G(\mathcal{V}, \mathcal{W}_m)$. The function *assignMessage* assigns the message to the route if there is available bandwidth. In case m is assigned, the path r_m^k is added to the routing set R_m for message m. Then, $\mathcal{M}_\psi$ is updated, i.e., if m reached the required redundancy, it is removed from $\mathcal{M}_\psi$.

3.1.2 Adaptive Weighting and Labeling

When the topology changes by adding a new route r_m to the routing set $\mathcal{R}$, the arcs weights $\mathcal{W}$ are updated to represent the next $\Delta_{i+1}\mathcal{C}$. The function *updateWeights* is responsible for $\mathcal{W}$ updating as shown in line 17 in Algorithm 1.

Algorithm 2: The function $updateWeights()$

Input : $\{\mathcal{W}, \mathcal{T}, r_m\}$
Output: $\{\mathcal{W}^*, \mathcal{T}^*\}$

1 $\bar{\mathcal{T}} \leftarrow activateElements(\mathcal{T}, r_m)$
2 $\mathcal{T}^* \leftarrow upgradeBridges(\bar{\mathcal{T}}, r_m)$
3 $\bar{\mathcal{L}} \leftarrow findOpenLinks(\mathcal{T}^*, r_m)$
4 $\forall l \in \bar{\mathcal{L}} : w_l^* \leftarrow \infty$
5 $\mathcal{W}^* \leftarrow updateSelectionCost(\mathcal{W}, \mathcal{T}^*, \mathcal{A})$

The main steps in the implementation of the *updateWeights* function are shown in Algorithm 2. The function *activateElements* finds and activates the bridges and links that are used for the first time. Then the function *upgradeBridges* finds and changes the type of the already active bridges that get an additional port in the current iteration. After that, the function *findOpenLinks* is responsible of finding the discarded arcs $\bar{\mathcal{L}}$ that cannot carry massages due to the limited number of ports

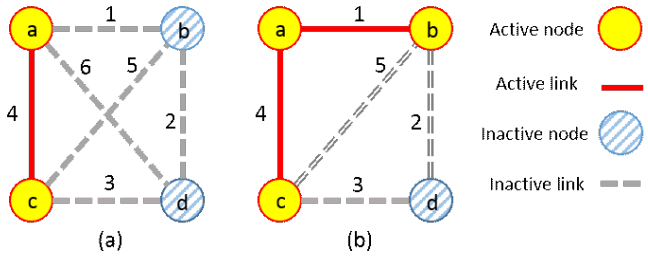


Figure 3.1: Example on weights updating.

supported by the bridges in $\mathcal{A}$. The weight of those arcs w_l^* are assigned to be ∞ which prevents using them in the next iteration. Finally, the function *updateSelectionCost* computes $\Delta_{i+1}\mathcal{C}_w$ for each arc $w \in \mathcal{W}$.

In order to illustrate the mechanism of the function *updateWeights*, consider the graph example depicted in Fig. 3.1. In particular, Fig. 3.1(a) shows the network status before updating the weights given that the added path in this iteration is $r_m = \langle a, b \rangle$ and the bridge type available in $\mathcal{A}$ supports only two ports. The function *activateElements*, in line 1, finds that constructing the path r_m implies activating bridge b and link l_1. Then, the function *findOpenLinks* adds the link l_6 to $\bar{\mathcal{L}}$ since bridge a cannot upgrade to support more than two ports according to $\mathcal{A}$. Hence, the arc that connects between node a and d is removed in line 3 i.e., $w_6^* = \infty$. The function *updateSelectionCost* changes the cost of selecting links $\{l_1, l_2, l_5\}$ to become as the following: $w(l_1) = \alpha$ since it is already part of the network. $w(l_2) = b_{cost} + l_{cost} + \alpha$, where b_{cost} and l_{cost} are the monetary cost of the link and the bridge since selecting l_2 in the next iteration implies activating the bridge d as well as the link l_2. Finally, $w(l_5) = l_{cost} + \alpha$ since both bridges b and c are already activated. Hence, selecting l_5 in the next iteration implies activating a single link.

3.2 Redundancy Using Multipath Routing

Adding new applications or modifying existing ones in the CPS implies updating the traffic planning for the underlying TSN, which has a fixed topology. In such cases, we introduce two reliability-aware routing techniques for AVB traffic using ILP-based formulation. The proposed techniques meet the transmission reliability of AVB traffic using spatial and temporal redundancy.

Most of the literature that addresses the routing problem considers non-redundant routing, i.e., the fault-tolerance requirements for the critical traffic are not considered (Mahfuzi et al., 2018a; Nayak, Duerr, & Rothermel, 2018; Schweissguth et al., 2017). Existing techniques assume that the network is physically replicated for fault-tolerance. On the other hand, we are interested in multipath routing to meet the required redundancy, which offers higher efficiency in terms of power, cost, and weight. In the latter approach, multiple replicas of the message are transmitted through disjoint paths, i.e., if one replica is corrupted, delayed, or dropped, the message is still received on time.

3.2.1 ILP Formulation

In the following, we introduce the redundancy and capacity constraints formulation of the proposed ILP-based Multiple-Path Routing (ILPMPR) technique. The ILPMPR jointly considers AVB and TT traffic to improve the design space exploration and allows further optimized solutions. The proposed ILP formulation can handle different redundancy constraints for each application. The formulated objective is minimizing the interference imposed on AVB messages from higher priority classes. The ILP problem is defined by the following constants:

- M_{TT}: The set of TT messages.

- M_{AVB}: The set of AVB messages.
- $M = M_{TT} \cup M_{AVB}$: The set of time-sensitive messages to be handle.
- $m \in M$: Index of messages.
- O_m: Number of redundant paths required for message m.
- P_m: Period of message m.
- N_m: The set of possible paths for message m.
- $n \in N_m$: Index of a possible path for message m.
- $L_{(m,n)}$: The set of links that compose path n for message m.
- $Q_{(m,n)}$: The number of links (hops) that compose path n for message m.
- Γ_l: Available bandwidth on link l.
- γ_m: Total bandwidth occupied by message m during one hyper-period.
- $T_{(m,l)}$: Transmission time for message m on link l.

The ILP uses the following variables:

- $X_{(m,n)}$: Binary variable indicates whether message $m \in M$ is transmitted through path n.
- $Xc_{(m,n)}$: Auxiliary binary variable complements $X_{(m,n)}$.
- $G_{(m,n,l)}$: Binary variable indicates whether message $m \in M$ passes through link l.
- D: Integer variable represents an upper bound for the WCD of M_{AVB} in μ sec.

The following constraints (16) - (6) define the valid routing. Each message should be assigned to a particular number O_m of disjoint paths. This requirement is guaranteed by constraint (2).

$$\forall m \in M : \sum_{n \in N_m} X_{(m,n)} = O_m \tag{2}$$

This constraint is applied to each message to enforce the ILP solver to select a specific number of redundant paths from the set N_m, representing possible paths between the source and destination nodes of the message m. The set N_m is obtained in advance using Yen's algorithm (Yen, 1970). N_m can include the whole possible paths for small instances. Generating all possible paths is a viable option for small-scale networks. However, it is not practical to consider all paths for large-scale networks such as Orion Crew Exploration Vehicle (CEV), which has 31 bridges (Paulitsch, Schmidt, Gstöttenbauer, Scherrer, & Kantz, 2011). We address this limitation using a heuristic approach by limiting N_m to the K-shortest paths.

We add constraints (3), (4), and (5) to define the variable $X_{(m,n)}$ as an indication that a certain path n is selected for message m. First, constraint (3) defines when $X_{(m,n)}$ is enforced to be '1'.

$$\forall m \in M, \forall n \in N_m : X_{(m,n)} + \left(Q_{(m,n)} - \sum_{l \in L_{(m,n)}} G_{(m,n,l)} \right) \geq 1 \tag{3}$$

This constraint states that when message m is assigned to all links of a path n, the value of $X_{(m,n)}$ is enforced to be '1' since the terms inside the brackets is equal to '0'. On the other hand, constraints (4) and (5) are introduced to specify when $X_{(m,n)}$ is enforced to be '0'.

$$\forall m \in M, \forall n \in N_m : -B \cdot Xc_{(m,n)} + \left(Q_{(m,n)} - \sum_{l \in L_{(m,n)}} G_{(m,n,l)} \right) \leq 0 \tag{4}$$

B is a big number that dominates other terms and deactivates the inequality when the message m is not routed through the path n. This constraint states that the value of $Xc_{(m,n)}$ is enforced to be '1' unless all links along path n are allocated for message m. in which $X_{(m,n)} = 1$ due to (5).

$$\forall m \in M, \forall n \in N_m : X_{(m,n)} + Xc_{(m,n)} = 1 \tag{5}$$

In order to ensure that the selected paths for each message are disjoint, we introduce the following constraint.

$$\forall m \in M, \forall l \in L : \sum_{n \in N_m} G_{(m,n,l)} \leq 1 \tag{6}$$

This constraint states that the redundant paths of message m are not allowed to use any common link.

Typical link capacity is in the range of 100 Mbps to 1 Gbps, where a portion of this capacity is dedicated to the high priority traffic (TT and AVB class-A) to prevent network starvation and serve the lower priority traffic. TSN standard allows maximum utilization of 75% of link capacity for high priority traffic (Yoo, Jo, Ju, & Park, 2017). Constraint (7) is added for each link in the network to ensure that the total size of the high priority traffic adheres to this specification.

$$\forall l \in L : \sum_{m \in M} \sum_{n' \in N_m^l} \left(G_{(m,n',l)} \cdot \gamma_m \right) \leq \Gamma_l \tag{7}$$

where $N_m^l \subset N_m$ is the set of possible paths for message m that use link l. The summations in this constraint represent the total size of TT and AVB traffic that passes through link l.

As mentioned earlier, we are interested in investigating the optimal routing for TT and AVB messages that minimizes WCD for AVB traffic. Given that variables $\mathbf{X}$ determine the routing of AVB and TT messages, and D_m refers to the maximum interference imposed on message m. Then, the objective is to minimize the maximum

D_m for $m \in M_{AVB}$ as depicted in (8).

$$\{X^*\} = \arg\min_{X} \left\{ \max_{m \in M_{AVB}} D_m \right\} \tag{8}$$

In order to define an upper bound for the maximum D_m for $m \in M_{AVB}$, we introduce the following constraint:

$$\forall m \in M_{AVB}, \forall n \in N_m : -B \cdot Xc_{mn} + \sum_{l \in L_{(m,n)}} \sum_{\substack{m' \in M \\ m' \neq m}} \left(G_{(m',n,l)} \cdot T_{(m,l)} \right) \leq D_m \tag{9}$$

The second term represents the total delay due to messages routed through all links along path n. Finally, minimizing the objective variable D_m generates the best routing solution.

3.3 Enhance Reliability By Temporal Redundancy

Safety standards such as ISO26262 standard state that safety-critical applications should receive -at least- one correct message every application-specific interval, namely, Diagnostic Test Interval (DTI), to be considered as correctly functioning (ISO 26262, 2011). The reliability of critical traffic is affected by the corruption of the message due to transient errors. Thus, error mitigation techniques should be used to enhance transmission reliability. The temporal redundancy can be employed by sending multiple copies of each frame through a single path at different times. Such technique enhance the soft reliability constraints, i.e., MTTF as shown in (Smirnov et al., 2016). The temporal redundancy approach imposes less resource-intensive constraints compared to the spatial redundancy. Such a low-cost approach satisfies

the reliability requirements for a broad spectrum of critical applications. Applying temporal redundancy as a separate step after the routing step is studied in (Smirnov et al., 2016). In this scenario, the redundancy is assigned for streams that already have fixed routing. We will call this strategy of performing two separate steps as Detached Temporal Redundant Routing (DTRR) approach. The DTRR neglects the interrelation between the messages routing and the redundancy assignment, limiting the design space exploration. For instance, the gain in the transmission reliability depends on the available bandwidth along the selected path. Routing a message through a larger bandwidth path allows more redundancy, which implies higher reliability. Furthermore, transmission reliability depends on the route itself, i.e., longer paths imply higher transient error rates. Moreover, DTRR approach may result in an infeasible redundancy assignment due to the bandwidth limitation, which implies utilizing more resources.

To ensure better design space exploration, we introduce a new technique to simultaneously investigate the messages routing and redundancy assignment in single-step using ILP solver. The proposed ILP-based Temporal Redundant Routing (ILPTRR) addresses the reliability requirements of moderate criticality traffic, especially for strict resource budget designs, which may restrict all messages' spatial redundancy. ILPTRR selects the route of each AVB message based on the available bandwidth taking into account the temporal redundancy overheads. To ensure scalability, we limit the search space to consider only the k-shortest paths. The proper value of k is specified according to the problem size.

3.3.1 Transmission Reliability Under Temporal Redundant Routing

Given that every message m has a specific DTI namely, (Td_m) and minimum Mean Time To Detect Error (MTTDE), (ψ_m). The value of Td_m as well as ψ_m is assumed to be predefined by the design engineers based on the criticality of the application. Whereas, r_m refers to the number of replicas of message m that are sent during Td_m. The value of r_m is determined such that ψ_m is satisfied for each message and the MTTDE for the whole network, $MTTDE_N$, is maximized. N_m represents the set of all possible paths for the message m. Transmitted messages are vulnerable to transient errors during every transmission through the network's links with a specific Bit Error Rate (BER). Given that the transmission reliability of each AVB stream is measured by its MTTDE, the probability P_{Cm} that a message $m_i \in M$ is delivered to its destination can be computed as Eq. (10).

$$P_{Cm} = (1 - BER)^{Nr_m \cdot S_m} \tag{10}$$

where Nr_m and S_m refer to the path length and the size of message m. Given that temporal redundancy is applied, a transmission failure for message m occurs if the network fails to deliver -at least- one correct replica of this message during Td_m. Hence, the probability of the transmission failure P_{Fm} is obtained as shown in Eq. (11).

$$P_{Fm} = (1 - P_{Cm})^{\gamma_m} \tag{11}$$

where γ_m is the minimum number of replicas which reach the destination within a interval of Td_m according to the maximum jitter which affects the message J_{max}. γ_m

is computed as follows:

$$\gamma_m = r_m - \frac{J_{max}}{Td_m} \tag{12}$$

Then, MTTDE for transmission of m can be computed by Eq. (13) which is introduced in (Smirnov et al., 2016).

$$MTTDE_m = \int_0^\infty (1 - P_{Fm})^{\frac{t}{Td_m}} dt = -\frac{Td_m}{\ln(1 - P_{Fm})} \tag{13}$$

Then, the failure rate of transmission for message m is computed by Eq. (14).

$$\lambda_m = \frac{1}{MTTDE_m} = -\frac{\ln(1 - P_{Fm})}{Td_m} \tag{14}$$

Finally, $MTTDE_N$, which is the objective function of the proposed routing algorithm, is computed by Eq. (15).

$$MTTDE_N = \frac{1}{\sum_{m \in M} \lambda_m} \tag{15}$$

3.3.2 ILP Formulation

In the following, the formulation for the valid routing constraints, capacity constraints, and reliability constraint. The ILP is defined by the following constraints:

- M_{AVB}: The set of AVB messages.
- N_m: The set of possible paths for message m.
- $L_{(m,n)}$: The set of links that compose path n for message m.
- C_l: Available capacity of link $l \in L$.
- $S_{(m,l)}$: Transmission time for message m on link l.
- $R_{(m,n)}$:Repetitions r_m that satisfy the required MTTDE for message m if it is routed through path n.

- $Q_{(m,n)}$: Length (number of hops) of the path n for message m.

The ILP uses the following decision variables:

- $Y_{(m,n,r)}$: Binary variable indicates whether message m is routed to path n_m with repetition r.
- $H_{(m,l,r)}$: Binary variable indicates whether message m passes through link l with repetition r.
- $Yc_{(m,n,r)}$: Auxiliary binary variable represents the complement of $Y_{(m,n,r)}$.

The routing is considered valid *iff* there is a particular path $p_m \in P$ is assigned for each message $m \in M_{AVB}$ with particular repetition $r_m \in R_m$. To enforce the algorithm to generate a valid routing, we introduce constraint (16) for every message:

$$\forall m \in M_{AVB} : \sum_{n \in N_m} \sum_{r \in R_{mn}} Y_{(m,n,r)} = 1 \tag{16}$$

This constraint ensures valid routing by enforcing one path to be assigned for every message $m \in M_{AVB}$. The value of variable $Y_{(m,n,r)}$ is defined by the following constraints (17) to (19). We introduce constraint (17) to enforce $Y_{(m,n,r)}$ to be '1' if the path n is assigned to message m with repetition r.

$$\forall m \in M_{AVB}, \forall n \in N_m, \forall r \in R_{(m,n)} : Y_{(m,n,r)} + \left(Q_{(m,n)} - \sum_{l \in L_{(m,n)}} H_{(m,l,r)} \right) \geq 1 \tag{17}$$

This constraint states that if a message m is assigned to all links that compose path n with repetition r, then the sum term over links $l \in L_{(m,n)}$ will be equal to the path length $Q_{(m,n)}$. Thus, $Y_{(m,n,r)}$ is enforced to be '1'. Whereas, constraints (18) and (19) are introduced to ensure that $Y_{(m,n,c)}$ cannot be activated unless all links along path n, $L_{(m,n)}$, are allocated for message m.

$$\forall m \in M_{AVB}, \forall n \in N_m, \forall r \in R_m : -N \cdot Yc_{(m,n,r)} + \left(Q_{(m,n)} - \sum_{l \in L_{(m,n)}} H_{(m,l,r)} \right) \leq 0 \tag{18}$$

This constraint states that a message m is not assigned to path n with repetition r unless all links $l \in L_{(m,n)}$ are assigned for message m with repetition r. $Yc_{(m,n,r)}$ is the complement of $Y_{(m,n,r)}$ as defined in constraint (19).

$$\forall m \in M_{AVB}, \forall n \in N_m, \forall r \in R_m : Y_{(m,n,r)} + Yc_{(m,n,r)} = 1 \tag{19}$$

Constraint (20) is introduced to represent the available capacity of physical links.

$$\forall l \in L : \sum_{m \in M_{AVB}} \sum_{r \in R_m} H_{(m,l,r)} \cdot S_{(m,l)} \cdot r \leq C_l \tag{20}$$

This constraint states that a link cannot be loaded by a traffic which exceeds its available capacity. To ensure that every message $m \in M_{AVB}$ has a particular repetition, constraint (21) is introduced.

$$\forall m \in M_{AVB}, \forall n \in N_m, \forall l \in L_{(m,n)} : \sum_{r \in R_{(m,n)}} H_{(m,l,r)} \leq 1 \tag{21}$$

This constraint states that message m cannot has different repetitions at any link $l \in L_{(m,n)}$.

3.4 Evaluation

3.4.1 Topology Synthesis

We evaluate the scalability of the RATS algorithm in terms of runtime with different numbers of ESs and messages. Furthermore, we show the efficiency of JTRSS for the topology cost and the scheduling feasibility comparing to two different approaches. In this analysis, 380 synthetic test cases are generated to perform the evaluation. MATLAB 2014a is employed to implement the algorithm. The reported results have been carried out on a workstation with an Intel Core i7 6820HQ processor

Table 3.3: Example Library for Bridges Modules

Module ID	Number of Ports	Cost
b_1	2	3
b_2	3	6
b_3	4	8

running at 3.0 GHz and 16 GB RAM.

Efficiency Evaluation

First, we evaluate the efficiency of the proposed algorithm by comparing it with two different approaches. The first approach is based on the NLR strategy, as described in (Annighoefer, Reif, & Thieleck, 2014). This approach realizes the multipath transmission by using multiple copies of the network topology. The second approach is based on the Separate-Synthesis (SS) strategy described in Section 4. This approach is influenced by the work introduced in (Gavrilut et al., 2017). To compare these approaches, we generate 80 synthetic test cases with various numbers of messages (30, 60, 90, and 120) randomly distributed among six ESs. All messages have a payload of 1,500 Bytes, a period of (1, 2, 3, and 10ms) and a required $TL = 2$. The link speed is 100 Mbps for all experiments, and the bridges library $\mathcal{A}$ in Table 3.3 is used.

The average cost and the percentage of feasible scheduling are depicted in Fig. 3.2. It can be noticed that the main drawback of the NLR approach is the higher cost. Nevertheless, some cases did not provide a feasible schedule since the routing and scheduling are not solved jointly. SS approach provides the minimum topology cost since it does not consider the feasibility of message scheduling. However, it suffers from a high percentage of the cases that lead to an infeasible schedule. In this case,

rerouting the traffic or even re-synthesize the topology by adding some new bridges and links may be required. On the other hand, the proposed JTRSS provides less cost than NLR approach while ensuring feasible scheduling.

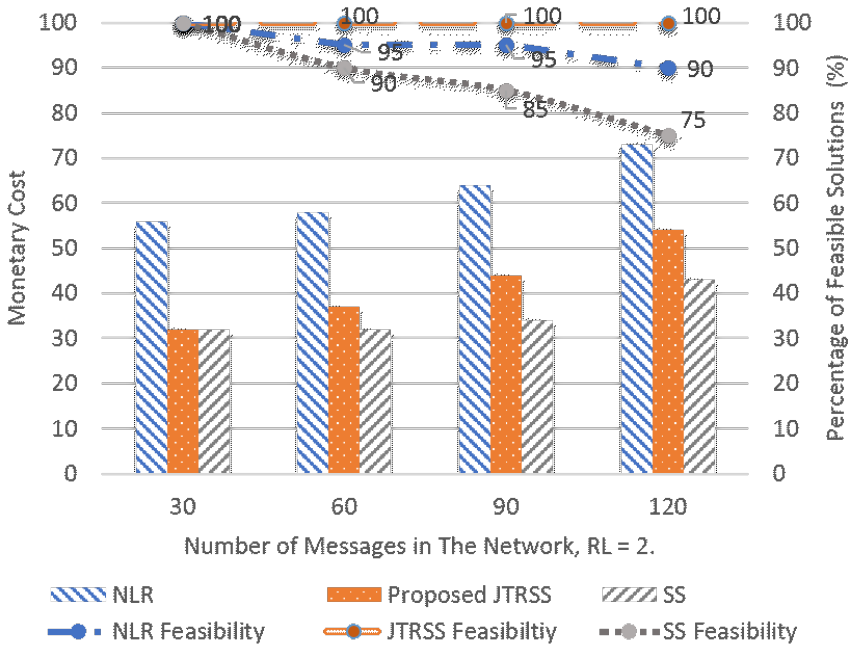
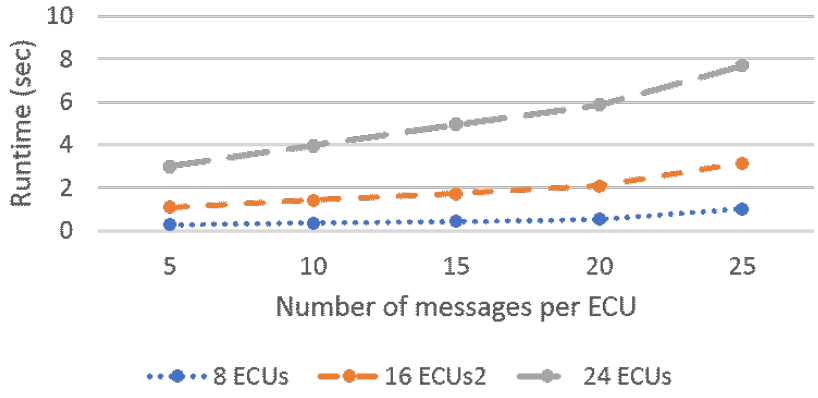


Figure 3.2: Average cost and the percentage of feasible scheduling for 80 synthetic test cases with different number of messages randomly distributed among 6 ESs.

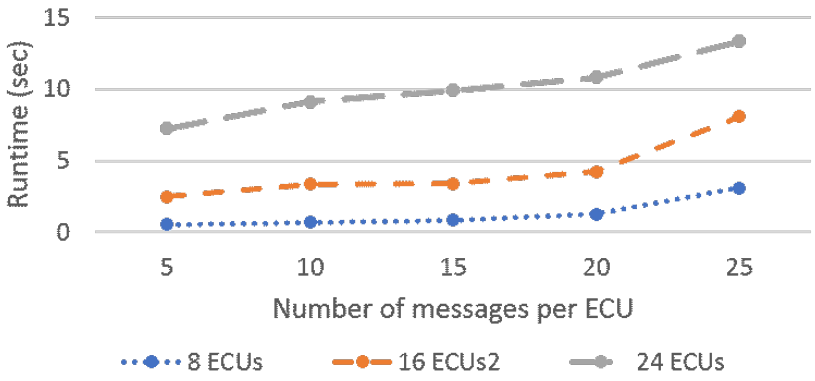
Scalability Evaluation

To evaluate the proposed approach's scalability, we used 300 synthetic test cases to cover the following settings. The problems are composed of 8, 16, and 24 ESs that exchange numbers of messages varied from 5 to 25 messages per ES. All messages have a payload of 1,500 Bytes, which is the Ethernet Maximum Transmission Unit (MTU), and a period of 1, 2, 3, and 10ms. At the same time, the required tolerance level is chosen to be 2 or 3. The source and destination of the messages are assigned randomly. The average runtime is depicted in Fig. 3.3. The results show the high scalability

of the proposed JTRSS algorithm such that it can handle large-scale problems, e.g., 24 ESs with 600 messages within a very short runtime about 8 seconds on average. Furthermore, the linear increase of runtime with respect to the number of ESs and the number of messages points to the potential of handling large-scale synthesis problems.



(a)



(b)

Figure 3.3: Average runtime for 300 synthetic test cases with varied numbers of ESs and messages.

3.4.2 Multipath Routing

Performance Evaluation

To evaluate the performance of the ILPMPR technique, we investigate the resultant WCD reduction comparing to the typical Shortest Path Routing (SPR) technique. Both techniques are implemented in the same framework. All test cases have been solved, and the resulting maximum WCD is shown in Fig. 3.4 with respect to their sizes. It can be observed from these results that the proposed technique has a significant impact, i.e., 40%, on the WCD comparing to the SPR technique. This impact increases with the number of messages. For instance, in the case of 60 messages, a reduction of 60% is achieved compared to the SPR technique.

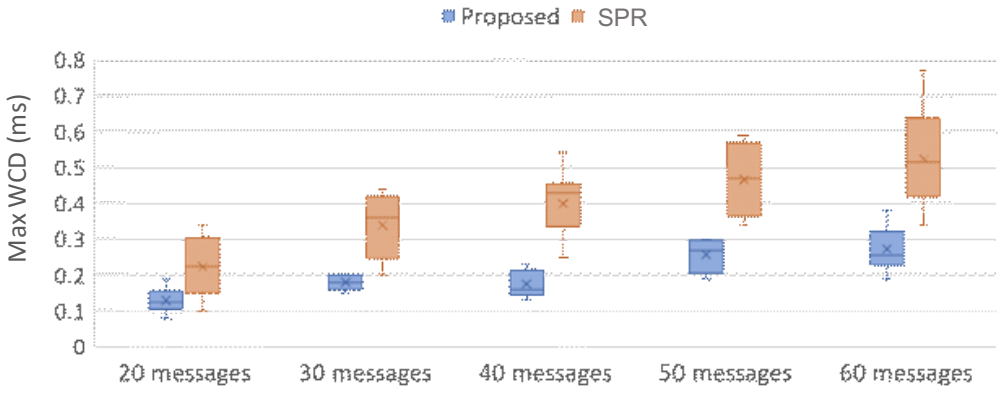


Figure 3.4: Worst case delay resulting using the proposed technique and the shortest path routing for different loaded networks

Case Study: Orion Crew Exploration Vehicle

Orion project is intended to be the next-generation Crew Exploration Vehicle (CEV) instead of the ended Space Shuttle Program (McCabe, Baggerman, & Verma,

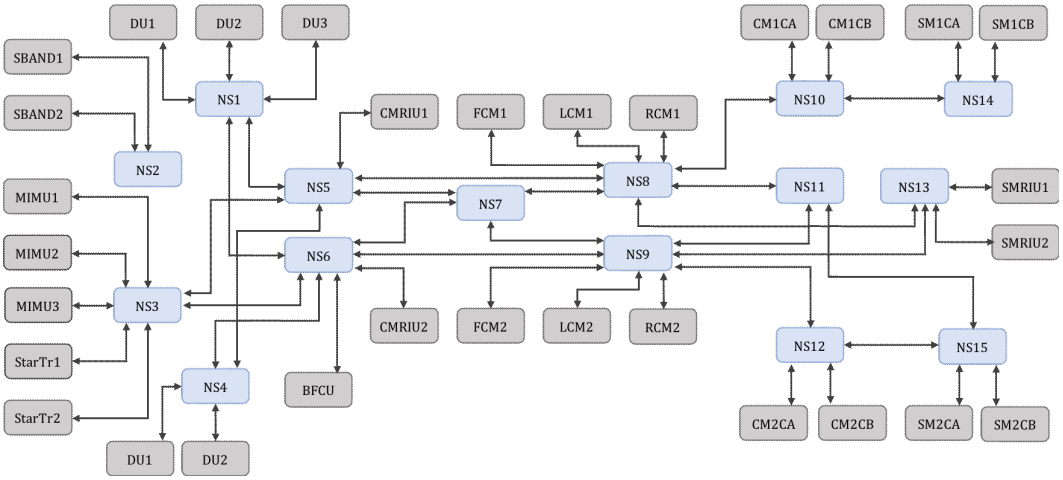


Figure 3.5: Network topology in the Orion CEV.

2009). Orion has strict reliability requirements, e.g., it has to meet an overall reliability allocation of 0.9999 for up to 5000 hours of continuous operation at a time under strict weight, and power constraints (Paulitsch et al., 2011). Concerning the communication requirements, Orion adapts an Ethernet-based switched communication network (Tămaş-Selicean et al., 2015). In this section, multipath routing is determined for two setups of 50 and 100 messages with the network topology and the specifications adopted in (Tămaş-Selicean et al., 2015) with RL = 2. The relative reduction in WCD of both setups using the proposed technique compared with the SPR technique is shown in Fig. 3.6.

Fig. 3.6 shows the reduction in WCD by considering different shortest path values. In particular, these results show that handling AVB and TT messages simultaneously by the proposed technique achieves a 30% reduction in 50 messages. Moreover, it shows a further reduction of up to 65% for the higher utilized case of 100 messages. On the other hand, results demonstrate that the proposed technique reduces WCD by increasing the number of considered paths, e.g., by considering the five shortest paths. This observation can be explained by the fact that larger K allows better

design space exploration, which improves the optimization results.

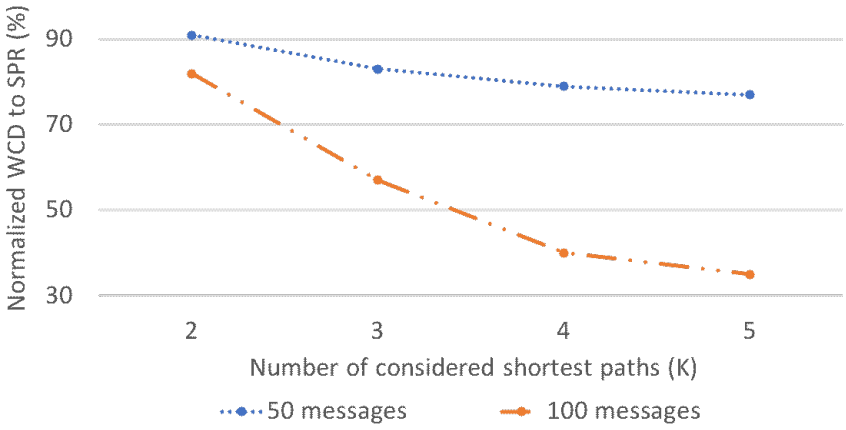


Figure 3.6: The relative value of WCD divided on the reference WCD value resulting by shortest path routing approach for different number of messages and different values of K.

To investigate the impact of the joint routing of AVB and TT traffic, the reduction in WCD when applying the proposed technique is compared with the 2-step routing technique. In 2-steps routing, the TT traffic is routed in advance. Afterward, the routing of AVB traffic is determined, such that WCD is minimized. The relative value of WCD divided on the reference WCD value resulting from a 2-step routing approach for a different number of messages, and different values of K are shown in Fig. 3.7. The proposed technique outperforms the 2-steps routing in both 50- and 100-messages cases for all K values. Moreover, it is observed that the advantage and the efficiency of the proposed technique become more evident when the high-loaded networks increase the impact of the proposed technique in reducing the AVB traffic delay. For instance, the proposed joint routing reduces the maximum WCD up to 42% for the case of 100 messages with K=5. These results highlight the importance of considering the timing requirements of non-scheduled traffic, e.g., AVB during TT traffic routing, to ensure optimal traffic planning.

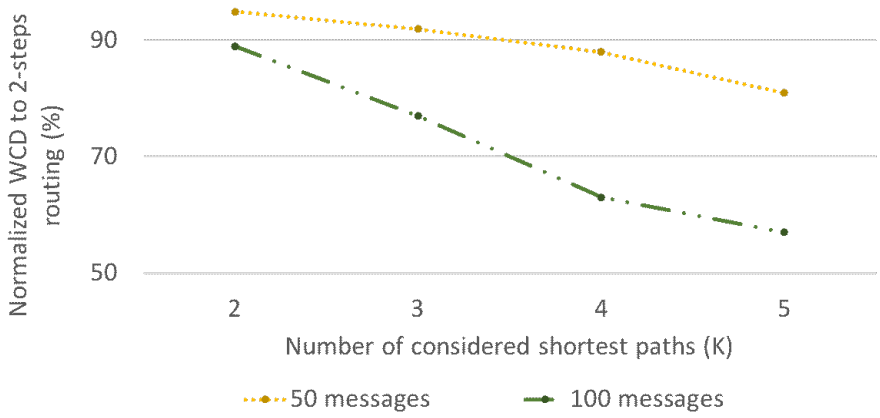


Figure 3.7: The relative value of WCD divided on the reference WCD value resulting by 2-steps routing approach for different number of messages and different values of K.

3.4.3 Temporal Redundancy

Performance Evaluation

The first analysis investigates the efficiency of the proposed ILP formulation in terms of the runtime. We present the runtime of 24 synthetic test cases of different sizes and loads. In particular, networks composed of 3, 4, and 5 switches, i.e., have paths length up to 2, 3, and 4 communication hops, respectively, are considered. A different number of messages are considered ranges from 20 to 90 messages are routed through each network. The messages' sources and destinations are randomly assigned from a set of 20 ES in each network. Messages have a payload of 1,500 Bytes, a period of 0.01 seconds, a required MTTDE of 10^6 second for BER of 10^{-9}, and a repetition of up to 3 replicas. Messages are transmitted over 100 Mbit Ethernet links. The maximum link utilization is chosen to be 50%. The runtime results for test cases are shown in Fig. 3.8.

We can see the super-linear growth of the solving time with the number of messages

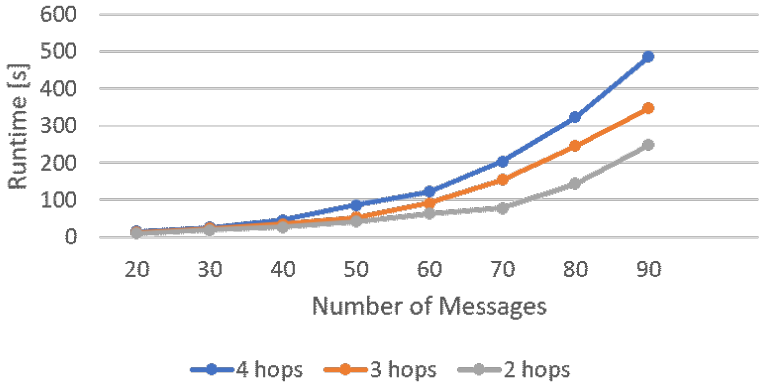


Figure 3.8: Runtime for 24 synthetic test cases for different number of hops and messages by the proposed reliability-aware routing.

and the size of the network. However, these values are still reasonable for off-line processing at the design stage.

Co-optimization Case Study

This section considers a network that consists of 13 ESs exchanging 70 messages over four switches. Messages are routed through paths of length up to 3 communication hops. Messages have payloads of 1, 2, and 3 KB, a period of 1, 2, 5, 10, and 20 ms, required MTTDE of 10^6 second for BER of 10^{-9}, and a repetition up to 3 replicas. Messages are transmitted over 100 Mbit Ethernet links. We solved the optimization problem for six different utilization constraints ranges from 25% to 75% of the bandwidth of the links to explore the trade-off between transmission reliability and network utilization. Fig. 3.9 depicts the optimal transmission reliability for different possible values of the maximum link utilization. This co-optimization yields 6 Pareto-optimal solutions. Increasing the available bandwidth implies that messages can be routed through shorter paths, and more replicas can be sent for each message. These factors explain the increase of the transmission reliability depicted in Fig. 3.9. The distribution of the 70 messages in terms of the number of assigned replicas for

different utilization constraints is shown in Fig. 3.10. It can be noticed that, by allocating more bandwidth, the number of messages with two repetitions is decreasing, while the number of messages that get the maximum repetitions (3 replicas in this case) is increasing.

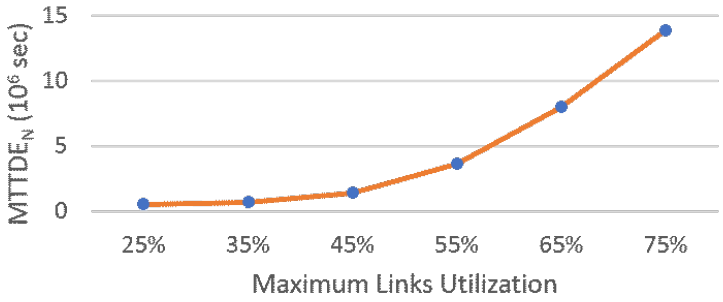


Figure 3.9: $MTTDE_N$ for Different Utilization Constraints.

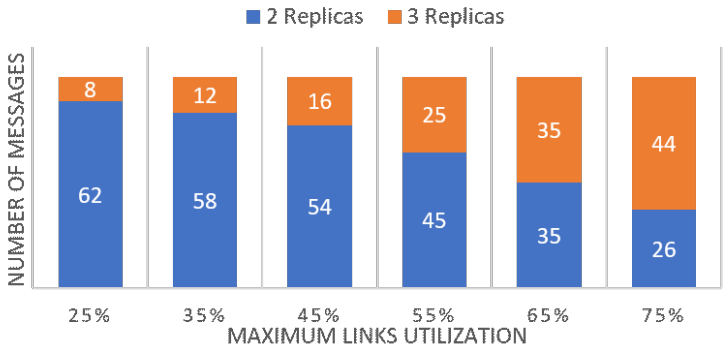


Figure 3.10: Messages Distribution With Respect to The Repetitions.

To demonstrate the advantage of the proposed approach over the 2-steps approaches, we tried to solve the same routing problem using *routing-first* and *reliability-first* approaches for different link utilization constraints varies from 25% to 75%. Table 3.4 shows the routing results regarding whether a feasible solution is found. We can notice that the proposed approach can find a solution with the utilization of 25% only. Whereas, *routing-first* and *reliability-first* approaches needed up to 65% and 45% of the link capacities, respectively, to find a solution.

Table 3.4: Feasibility using the proposed approach, *routing-first*, and *reliability-first*.

U_{max}	15%	25%	35%	45%	55%	65%	75%
Proposed ILPMPR	✗	✓	✓	✓	✓	✓	✓
Routing-first	✗	✗	✗	✓	✓	✓	✓
Reliability-first	✗	✗	✗	✗	✗	✓	✓

Chapter 4

Article I: Routing and Scheduling of Time-Triggered Traffic in Time Sensitive Networks

Authors: Ayman Atallah, Ghaith Bany Hamad, Otmane Ait Mohamed

Abstract: This paper addresses the following research question: *how to compute no-wait schedules and multipath routings for large-scale Time Sensitive Networks (TSNs)?* TSN must guarantee low latency and fault tolerance. The former requirement is achieved by sending the messages according to a no-wait schedule, while the latter is achieved by routing each message through multiple streams of disjoint paths. Computing such schedule and routing is an NP-hard problem. In this work, the above question is addressed by a three-fold solution: (i) An Iterated ILP-based Scheduling (IIS) technique for scalability; (ii) The Degree of Conflict (DoC) between the IIS iterations is minimized by the DoC-Aware Streams Partitioning (DASP) technique which improves the success rate of the IIS; (iii) The fault-tolerance is guaranteed by

a DoC-Aware Multipath Routing (DAMR) technique which integrates the DASP for further improvement in the success rate. Two hundred synthetic test cases are used for performance evaluation. The proposed method scales well, i.e., it handled networks of 21 bridges and 480 messages under 40 minutes timeout. The success rate of the highly utilized instances raised from 47% by Random Streams Partitioning (RSP) to 90% by the proposed method.

4.1 Introduction

The deployment of CPSs as a replacement for the typical mechanical components is increasing in the industrial automation systems (Da Xu, He, & Li, 2014). This transition requires an efficient integration between the computing units and the controlled physical processes. Such integration demands reliable and real-time communication to deliver control and feedback messages (Poovendran, 2010). The complexity escalation due to the industrial revolution, i.e., *Industry 4.0*, hikes the design constraints for communication networks in terms of bandwidth, coverage, and cost (Finn, 2018). Ethernet is an attractive networking solution in terms of scalability and cost. Nonetheless, affording deterministic low latency communication over Ethernet architecture is hard due to several inherent limitations, such as the non-deterministic queuing delay and packets collisions (Decotignie, 2005).

IEEE 802.1 standard is introduced to address these limitations by a new Ethernet-based solution called TSN (Messenger, 2018). TSN supports real-time applications with zero packet loss due to buffer congestion and bounded end-to-end latency (Finn, 2018). Moreover, TSN allows the coexistence of time-critical and best effort traffic on the same network which delivers cost savings and reserves the backward compatibility. A message is exchanged in TSN as a stream of frames transmitted from one source

to one or more destinations with specific path, size, and priority. TSN also allows a mixed-criticality communication of TT class which adhere to static schedules, AVB class, and BE streams (Laursen, Pop, & Steiner, 2016). The TT stream is periodic and has a deterministic delivery of low latency, e.g., tens of microseconds (Steiner, Craciunas, & Oliver, 2018). The delivery of AVB streams has a bounded WCD, e.g., less than 2 *ms* for AVB class-A. On the other hand, BE class has no timing guarantees.

To enable the time-aware scheduling defined in 802.1Q, TSN employs precise time synchronization protocols like 802.1AS (Nayak, Dürr, & Rothermel, 2018). Such protocols allow all bridges and ESs in the network to synchronize their local clocks by less than 1 microsecond precision (Finn, 2018). In particular, each TT frame is transmitted according to a static schedule which allows deterministic communication with bounded jitter and end-to-end latency (Steiner et al., 2018). The static schedule also prevents the interference from lower priority classes by preempting them in advance of the TT frames arrival. For some critical systems with stringent requirements, the static schedule can provide jitter-free transmission and deterministic end-to-end latency guarantees by isolating TT frames in every egress port as introduced in (Craciunas et al., 2016). On top of the timing requirements, the delivery of TT messages must tolerate frames corruption and links failure instantaneously. This makes the on-demand packet retransmission and network reconfiguration techniques inapplicable for TSN. Hence, the Frame Replication and Elimination for Reliability (FRER) defined in 802.1CB is introduced for fault-tolerance. The FRER uses multiple disjoint paths to deliver the message to tolerate any failure of a single link in the network (Finn, 2018). The number of disjoint paths, namely, Redundancy Level (RL) can be specified according to certification standards, such as IEC 61508 for the industrial applications.

Unlike the hardware and transmission functionalities, the routing and scheduling techniques have been left out of IEEE 802.1 standard (Steiner et al., 2018). Regarding the TSN scheduling, several formulations have been introduced to meet the timing requirements using Integer Linear Programming (ILP) (Lukasiewycz, Schneider, Goswami, & Chakraborty, 2012; Schweissguth et al., 2017) and Satisfiability Modulo Theories (SMT) (Craciunas et al., 2016). Nonetheless, such a scheduling task is an NP-hard problem (Tindell, Burns, & Wellings, 1992), and thus, the above techniques can only tackle small instances (Falk, Dürr, & Rothermel, 2018). The iterated scheduling, for scalability, is addressed by (Mahfuzi et al., 2018b) using the RSP. However, the performance of iterated scheduling is sensitive to the Degree of Conflict (DoC) across the stream groups (Pozo, Rodriguez-Navas, Hansson, & Steiner, 2017). Thus, the RSP, which neglects this conflict, results in a low success rate. Regarding the TSN routing, meeting the RL over multipath routing is typically ignored in the literature (Lukasiewycz et al., 2012; Mahfuzi et al., 2018b; Nayak, Duerr, & Rothermel, 2018; Schweissguth et al., 2017). Instead, many techniques assume a replication of the entire network for fault-tolerance (Tămaş-Selicean et al., 2015). This approach is inefficient in terms of cost, power, and area (Gavrilut et al., 2017), which is challenging in the resources-critical industrial applications.

In this paper, we address the following research question: *how to compute no-wait schedules and multipath routing for large-scale TSNs?* The proposed DA/IRS addresses this question by the following three techniques: (i) The Iterated ILP-based Scheduling (IIS) in order to attain high scheduling scalability by dividing the set of TT streams into multiple groups each of which is incrementally added to the schedule; (ii) A new graph based DASP in order to improve the success rate of the IIS; (iii) The DoC-Aware Multipath Routing (DAMR) in order to achieve fault-tolerance. The DAMR implement a multi-start iterated greedy heuristics to integrate

the DASP with the routing synthesis. A set of 200 synthetic test cases are used for the performance evaluation. The DA/IRS shows a good scalability and success rate, i.e., it handled networks of 21 bridges under a 40-minute time limit. Regarding the number of messages, it handled 480 messages under the same time limit. Furthermore, the proposed method has increased the success rate by 70% compared to the RSP.

The remainder of the paper is outlined as follows: Section II discusses the related work. The system model under consideration is described in Section III. Section 4.4 explains the DASP technique. Section 4.5 describes the DAMR technique, while the IIS technique is described in Section 4.6. Section 4.7 discusses the experimental results of the proposed method compared to state-of-the-art techniques using several test cases. The conclusion is presented in Section 4.8.

4.2 Related Work

The formal-based scheduling methods of TSN IEEE 802.1Qbv specifications are addressed in (Craciunas, Oliver, & AG, 2017; Craciunas et al., 2016) using an SMT formulation. Several ILP formulations for the joint routing and scheduling problem are introduced in (Falk et al., 2018; Schweissguth et al., 2017; Smirnov et al., 2017b) to address the strong interconnection between both tasks. An SMT-based iterative approach for joint routing and scheduling is introduced in (Mahfuzi et al., 2018b). Each iteration schedules a subset of frames which are determined based on a temporal slicing without ensuring a zero jitter between frames of the same stream. The iterated scheduling using RSP has been addressed by SMT formulation (Pozo, Steiner, Rodriguez-Navas, & Hansson, 2015). Whereas, an ILP formulation for the dynamic routing and scheduling problem is introduced in (Nayak, Dürr, & Rothermel, 2018) for the Time-Sensitive Software-Defined Networks (TSSDN). The no-wait scheduling

problem is addressed in (Dürr & Nayak, 2016) which avoids the frames jitter by assuming that all streams have the same period cycle, and thus the problem is restricted to one frame per stream. Nonetheless, a wide spectrum of CPSs exchange the messages in different frequencies. The FRER for fault-tolerance is typically neglected in the literature (Lukasiewycz et al., 2012; Mahfuzi et al., 2018b; Nayak, Duerr, & Rothermel, 2018; Schweissguth et al., 2017). On the other hand, the temporal redundancy has been investigated for bus-based network, i.e., FlexRay (Tanasa, Bordoloi, Eles, & Peng, 2010), and for Rate Constrained (RC) traffic in TSN (Atallah, Bany Hamad, & Ait Mohamed, 2018; Smirnov et al., 2016) to improve the MTTF of the overall network. The spatial redundancy is addressed in (Atallah et al., 2018; Gavrilut et al., 2017) in the context of topology synthesis. Whereas, the multipath routing for selected streams is addressed in (Smirnov et al., 2018) to improve the overall network reliability based on a best effort strategy.

4.3 System Model

4.3.1 Architecture Model

We consider an architecture model that represents a set of ESs denoted by **ES** connected via a set of bridges denoted by **B**. All physical links are Ethernet-based full-duplex links, and may have various speed, e.g., 100 Mbps, or 1 Gbps. **ES** and **B** are assumed precisely synchronized, i.e., they are capable of adhering with the computed schedules reasonably. Each egress port deploys a Time Aware Shaper (TAS) which adopts a First-In First-Out (FIFO) queuing paradigm while isolating the different traffic classes or priorities in one or more separate queues. Hence, the transmission order of frames of same priority depends on the arrival order, i.e., cannot

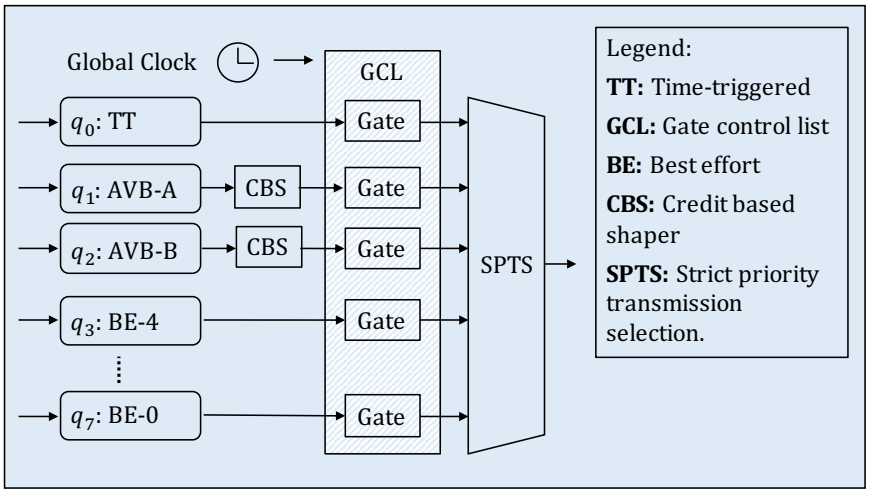


Figure 4.1: Egress port with IEEE 802.1Qbv enhancements.

be switched like TTEthernet network (Steiner et al., 2018). A simplified schematic of the TAS is shown in Fig. 1. The buffered frames are distributed on queues based on three Priority Code Point (PCP) bits in their headers. Each queue ends by a timed gate controlled by a Gate Control List (GCL). Every row in the GCL indicates the open and close gates for a certain time window starting from a relative time-stamp (Nayak, Duerr, & Rothermel, 2018). The GCL is periodically repeated every pre-defined cycle. Consequently, the TT schedule in TSN is synthesized on the queue level which affect all streams share the same queue (Craciunas et al., 2016). In other words, the static schedule of the TT streams is translated to the level of traffic classes as periodic transmission window (Craciunas et al., 2017). Among the open queues, the transmission selection is based on a strict priority scheme.

The network topology is modeled as a directed graph $\mathbf{G} \equiv (\mathbf{V}, \mathbf{L})$ where $\mathbf{V} = \mathbf{ES} \cup \mathbf{B}$ is the set of all communicating nodes, and the set $L \subseteq V \times V$ includes all directional links that connect any two nodes. An example of a network topology of $\mathbf{ES} = \{es_0, es_1, es_2, es_3\}$, and $\mathbf{B} = \{b_0, b_1, b_2, b_3, b_4, b_5\}$ is shown in Fig. 4.2. Each *full-duplex* link between nodes $(v_i, v_j) \in \mathbf{V}$ is considered as two separate directional links

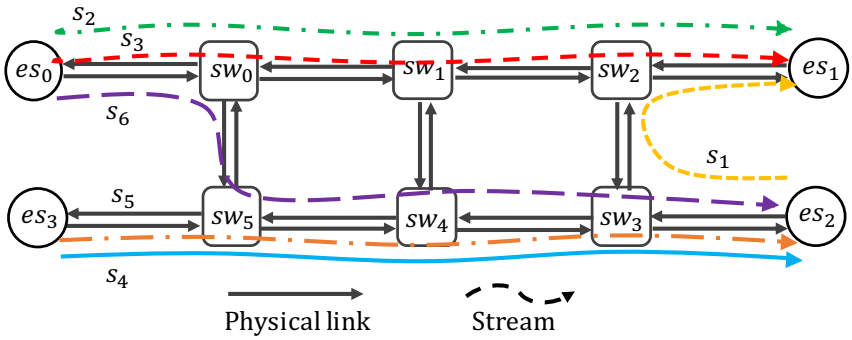


Figure 4.2: An illustrative example of four ESs (es_0, es_1, es_2, es_3) which exchange six streams $(s_0, s_1, s_2, s_3, s_4, s_5)$ via TSN network composed of six bridges $(b_0, b_1, b_2, b_3, b_4, b_5)$.

denoted by ordered pairs $[v_i, v_j]$ and $[v_j, v_i]$ where the first and the second elements refer to the sender and receiver nodes, respectively. A path $\mathcal{P}$ is an ordered sequence of physically connected nodes starting at the source v_s and ending at destination v_d where $v_s \in \mathbf{ES}$ and $v_d \in \mathbf{ES}$. For example, one of the paths between es_0 and es_1 is $\mathcal{P}^i_{(es_0,es_1)} = \{[es_0, b_0, b_1, b_2, es_1]\}$.

4.3.2 Application Model

We consider a set of time critical applications communicating over a TSN. Each application is represented by a periodic TT message m. The set of TT messages is defined by the tuple $\mathbf{M} \equiv (Sr, D, P, Si, Dl, Rl)$ where $Sr \equiv \{sr_m \in ES\}$ and $D \equiv \{d_m \in ES\}$ are the senders and the receivers of messages $m \in \mathbf{M}$. The frame size and period of the messages are defined by $Si \equiv \{si_m \in \mathbb{Z}^+\}$ and $P \equiv \{p_m \in \mathbb{Z}^+\}$, respectively. The maximum frame size, namely, Maximum Transmission Unit (MTU) is 1500 bytes which is the limit for Ethernet frame. Several frames of message m can be transmitted during the hyper-period which is the least common multiple of all messages' periods. The message deadline is defined by $Dl \equiv \{dl_m \in \mathbb{R}^+\}$. Finally, $Rl \equiv \{rl_m \in \mathbb{Z}^+\}$ is the given redundancy level of the messages $\mathbf{M}$. Each frame

is replicated at the first bridge after the source, while the replicas are eliminated at the last bridge before the destination. A stream $s_m^r \equiv (sr_m, d_m, p_m, si_m, \mathcal{P}_m^r)$ where $0 \leq r \leq rl_m$ distinguishes the frames that belong to message $m \in \mathbf{M}$ and flow through a specific path $\mathcal{P}_m^r$. In TSN, a TT stream can comprises multiple frames unlike TTEthernet which limits the streams to one frame with maximum size of MTU (Craciunas et al., 2016). The considered TT streams must be transmitted based on a no-wait schedule which is defined as follows:

Definition 1. *Let $ts_{(i,j)}$ and $te_{(i,j)}$ are the points of time at which the TT frame i starts and ends being transmitted over the hop j, and Ot_j is the total forwarding overhead time on hop j including processing and switching time, but excluding the queuing delay. Then, the no-wait schedule shall satisfy the following constraint:*

$$ts_{(i,j+1)} \leq te_{(i,j)} + Ot_{j+1} + \sigma \tag{22}$$

where $j+1$ and σ are the next hop in the path of frame i, and the macro tick of the network, respectively.

Since the solution validity is restricted by the no-wait constraint, the latency $L(s_m)$ of stream s_m routed over identical hops can be approximated as (23).

$$L(s_m) \approx |\mathcal{P}_m| \times (si_m + Ot) \tag{23}$$

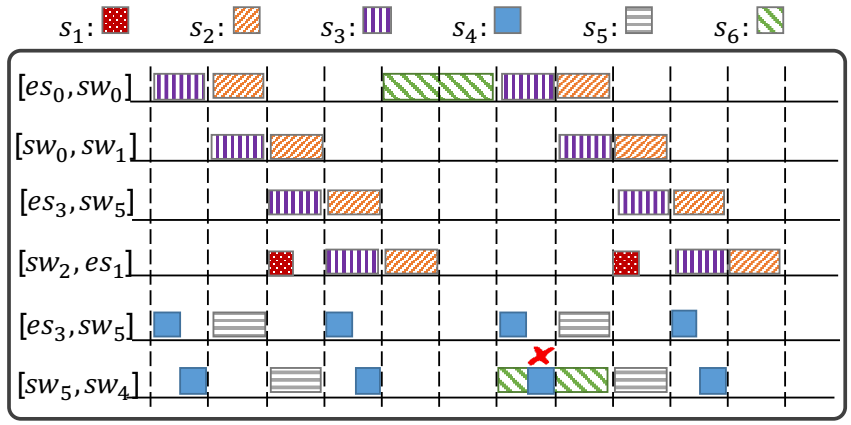
This constraint ensures a very low latency, e.g., in 1 Gbps network, the latency of the MTU over four hops is around 50 microseconds. In this work, we assume that $dl_m < L(s_m)$, and consequently, any valid solution meets the streams' deadlines. Without loss of generality, in this paper, the streams are considered to be unicast. However, multicast streams can be denoted as a set of unicast streams. In this paper,

we assume densely connected typologies, such that the possible paths between nodes $(v_i, v_j) \in \mathbf{ES}$ outnumbers the required RL for messages exchanged between these nodes. The set of all streams in the TSN is $\mathbf{S} \equiv \{s_m^r | \, m \in \mathbf{M}, 0 \leq r \leq rl_m\}$.

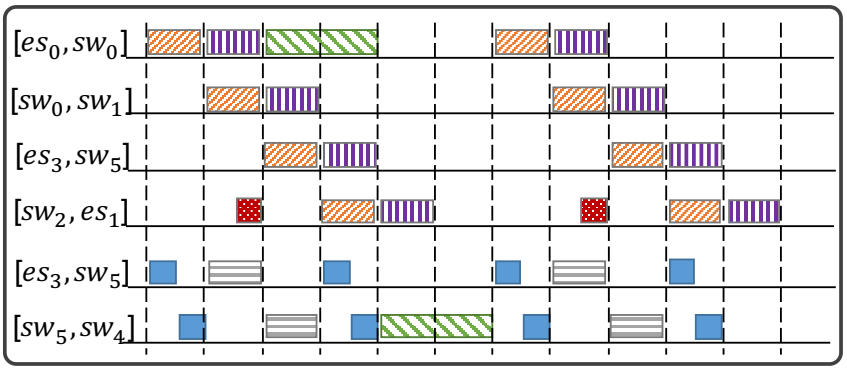
4.4 Graph-based Stream partitioning

To demonstrate the impact of the RSP on the iterated scheduling, let us consider the illustrative example shown in Fig. 4.2. Although TSN bandwidth is 100 Mbps or higher, in this example we assume 1 Mbps speed for all full-duplex links in the network. This assumption stretches the frames and allows to-scale visualization of the traffic in the hyper-period as shown in Fig. 4.3. The network carries a set of streams $\mathbf{S} = \{s_1, s_2, s_3, s_4, s_5, s_6\}$ which corresponds a set of six messages $\mathbf{M}$ with $Sr = \{es_2, es_0, es_0, es_3, es_3, es_0\}$, $D = \{es_1, es_1, es_1, es_2, es_2, es_2\}$, $P = \{2, 2, 2, 1, 2, 3\}$ ms, $Si = \{20, 40, 40, 20, 40, 80\}$ Bytes, and $Rl = \{1, 1, 1, 1, 1, 1\}$. The paths of $\mathbf{S}$ are visualized in Fig. 4.2. A no-wait schedule for $\mathbf{S}$ is required over two iterations. Let $\mathbf{S}$ is divided into two groups A, and B, i.e., $A \cup B = \mathbf{S}$ and $A \cap B = \emptyset$. Let the notation A/B refers to the order of handled groups starting from the left. Now, let us consider a random partitioning, i.e., $A1 = \{s_3, s_5, s_6\}$, and $B1 = \{s_1, s_2, s_4\}$. The Gantt chart shown in Fig. 4.3(a) visualizes the resulting schedule for $A1/B1$ over a timeline from 0 to 4 ms for the links of interest. Fig. 4.3(a) shows that the no-wait schedule is infeasible. Please note the irresolvable overlap between the third frame of s_4 and the first frame of s_6. This happened because streams s_5 and s_6 have been scheduled before s_4 which shares the link $[b_5, b_4]$ with them. To reduce the potential conflict between groups, we define the following dependency measure between streams.

Definition 2. *A Degree of Conflict (DoC) is a measure for the mutual dependency between streams. Such that, given two streams (s_i, s_j), the DoC between these streams*



(a)



(b)

Figure 4.3: The iterated no-wait scheduling of the illustrative example based on: (a) the RSP leading to an infeasible solution by handling $A1 = \{s_3, s_5, s_6\}$, then $B1 = \{s_1, s_2, s_4\}$; (b) the DASP leading to a feasible solution by handling $A3 = \{s_4, s_5, s_6\}$, then $B3 = \{s_1, s_2, s_3\}$.

$\mathcal{D}_{(i,j)}$ *is computed by* (24).

$$\mathcal{D}(i,j) = |\mathcal{P}_i \cap \mathcal{P}_j| \times \frac{si_i \times si_j}{p_i \times p_j} \tag{24}$$

The DoC encounters three conflict indications between the streams, namely the number of shared links, the frame size and frame period. For example, larger and more frequent frames are harder to be allocated later. Instead of RSP which neglects the DoC between streams, we propose a new streams partitioning technique,

namely DASP, to reduce DoC and improve the design space exploration of the iterated scheduling. The DASP technique generates stream groups based on the streams' paths, sizes, and periods, i.e., the streams with higher DoC tend to be grouped together. Such partition allows the ILP solver to better explore the feasible schedules for such streams.

The DASP technique (Algorithm 1) represents the TT streams as undirected graph $\mathcal{G}(\mathbf{S}, \mathbf{W})$. Every stream $s_i \in \mathbf{S}$ is represented as a node while the set $\mathbf{W} \subseteq \mathbf{S} \times \mathbf{S}$ represents the arc weight $w(i,j) = \mathcal{D}(i,j)$ between any two streams s_i, s_j. To explain the mapping of streams into a graph, let us revisit the illustrative example. The graph representation of the stream set $\mathcal{G}(S, W)$ is shown in Fig. 4.4 in which every node pair (s_i, s_j) that share some links are connected by an arc with a certain weight. A missing arc stands for a zero-weight arc between streams with no common links. The total potential conflict between groups is defined as follows.

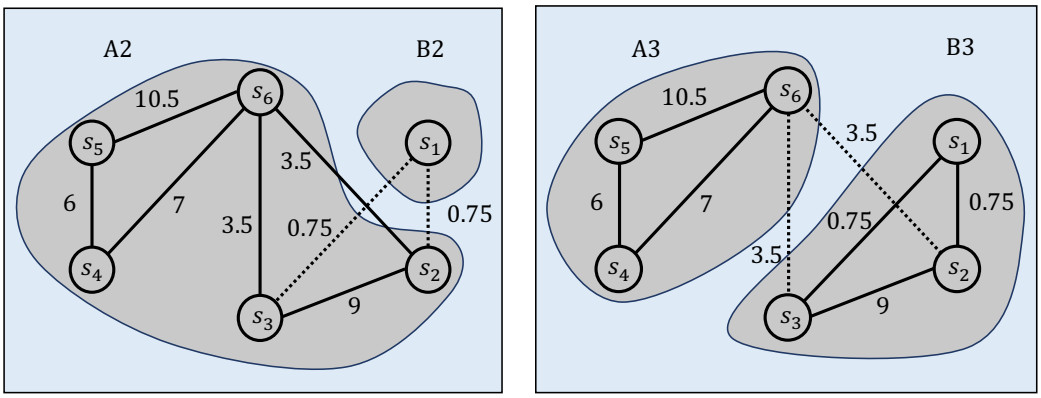
Algorithm 3: DoC Aware Stream Partitioning

Input : Stream Set $\mathbf{S}$,
Number of groups N_G
Output: Stream groups $S^n, n = 1, ..., N_G$,
Normalized cross-group conflict $nCGC$

1 Create the graph $\mathcal{G}$ based in $\mathbf{S}$
2 Compute $w_{(i,j)}, \forall (i,j) \in \mathbf{S}$ based on (24)
3 $(S^n, ncgc) \leftarrow ComputeNCut(W, N_G)$

Definition 3. *A CGC is a measure for the potential conflict between stream groups. Such that, given the graph $\mathcal{G}(\mathbf{S}, \mathbf{W})$, and n disjoint sets $g_1, g_2, ..., g_{N_G}$ where $\bigcup_{i=1}^{n} g_i = \mathbf{S}$, and $\bigcap_{i=1}^{n} g_i = \emptyset$, the CGC is the total weight of the arcs that connect nodes of different sets as computed in (25).*

$$cgc(g_1, g_2, ..., g_{N_G}) = \sum_{\substack{u \in g_i, v \in g_j \\ i \neq j}} w(u,v) \tag{25}$$



(a) (b)

Figure 4.4: A graph partition for the six streams (nodes) in the illustrative example at K = 2 by minimizing (a) the CGC which results in unbalanced groups; (b) the nCGC which results in balanced groups.

The partition of $\mathcal{G}$ which minimizes the CGC can be obtained by the min-cut algorithm introduced in (Wu & Leahy, 1993). However, minimizing CGC tends to generate one large set beside other small sets of isolated nodes (Wu & Leahy, 1993). For instance, the minimum CGC in our example results in unbalanced groups $A2 = \{s_2, s_3, s_4, s_5, s_6\}$ and $B2 = \{s_1\}$ as shown in Fig. 4.4(a). The size of the largest group $A2$ is close to the size of the original set $\mathbf{S}$ which shall limit the scalability of the iterated scheduling approach.

To generate groups of converged sizes, we adapt the Normalized Cut (NCut) framework (Shi & Malik, 2000) which is a well-known framework in the context of image processing. The NCut is used to generate a homogeneous and balanced pixels groups of the processed image (Shi & Malik, 2000). In our case, the NCut framework minimizes what we call the normalized CGC (nCGC) which is computed based on the NCut criteria as follows.

$$ncgc(g_1, g_2, ..., g_{N_G}) = \sum_{i=1}^{n} \frac{cgc(g_1, g_2, ..., g_{N_G})}{assoc(g_i, \mathbf{S})} \tag{26}$$

where

$$assoc(g_i, \mathbf{S}) = \sum_{u \in g_i, t \in \mathbf{S}} w(u,t) \tag{27}$$

Using the nCGC, the small sets of isolated nodes are no longer deliver the optimal partition since they will decrease the *assoc*. This prevents the biasing toward generating smaller sets (Shi & Malik, 2000). The optimal partition of the minimum nCGC is obtained by the NCut implementation in (Cour, Yu, & Shi, Copyright 2004). Revisiting our example in Fig. 4.2, the DASP results in two equal-sized groups $A3 = \{s_4, s_5, s_5\}$ and $B3 = \{s_1, s_2, s_3\}$ as depicted in Fig. 4.4(b). Whereas, a feasible no-wait schedule resulting by $A3/B3$ is shown in Fig. 4.3(b).

4.5 DoC-Aware Multipath routing

Given the TSN topology $\mathbf{G}$, and the TT messages $\mathbf{M}$, the DAMR technique generates an optimized stream set $\mathbf{S}$ that satisfies the RL of $\mathbf{M}$, while yielding a low nCGC. The DAMR comprises of three steps, namely the preprocessing, initial solutions construction, and local search for solutions refinement. The preprocessing step preserves the RL of each message $m \in \mathbf{M}$ by generating multiple sets of paths between the source and destination of m, namely Disjoint Routing Sets (DRSs).

Definition 4. *A Disjoint Routing Set of a message m denoted as γ_m is a number equal to rl_m of disjoint paths $\mathcal{P}_m^r$ that start form sr_m and end at d_m, such that $\gamma_m = \left\{ \mathcal{P}_m^0, ..., \mathcal{P}_m^{rl_m - 1} \middle| \mathcal{P}_m^i \cap \mathcal{P}_m^j = \{sr_m, b_{first}, b_{last}, d_m\} \right\}$ where b_{first} and b_{last} are the first and last bridges, respectively.*

For example, let us consider the network topology shown in Fig. 4.2 and a message m with $sr_m = es_0$, $d_m = es_3$, and $rl_m = 2$. There are three possible paths for m which are $\mathcal{P}_m^0 = \{[se_0, b_0, b_5, es_3]\}$, $\mathcal{P}_m^1 = \{[es_0, b_5, b_1, b_4, b_5, es_3]\}$, and $\mathcal{P}_m^2 =$

$\{[es_0, b_0, b_1, b_2, b_3, b_4, b_5, es_3]\}$. Then, m has two DRSs which are $\gamma_m^0 = \{\mathcal{P}_m^0, \mathcal{P}_m^1\}$ and $\gamma_m^1 = \{\mathcal{P}_m^0, \mathcal{P}_m^2\}$. In large-scale densely connected networks, the complete set of DRSs can be very large. Therefore, the preprocessing step generates a truncated set of DRSs defined as follows:

Definition 5. *A Truncated Multipath Routing Space (TMRS)* $\Gamma \equiv \{\gamma_m^j \,|\, m \in \mathbf{M}, 0 \leq j < j_{max}\}$ *is a set of DRSs that includes one up to* j_{max} *DRSs for each message* $m \in \mathbf{M}$.

Given a random vector $J \equiv \{J(m) \in [0, j_{max}) | \, m \in \mathbf{M}$, the definition of TMRS implies that J determines a complete set of streams for $\mathbf{M}$, namely $\mathbf{S}_J \equiv \left\{ \gamma_m^{J(m)} | \, m \in \mathbf{M} \right\}$. The objective of DAMR is to find the optimal streams set $\mathbf{S}_J^*$, such that:

$$\mathbf{S}^* = \arg\min_J cost(\mathbf{S}_J, N_G) \tag{28}$$

where $cost(S_J)$ is the nCGC computed by the DASP described in Section 4.4.

DAMR Procedures – The procedures of the multi-start iterated greedy heuristic for the DAMR are presented in Algorithm 4 as follows. In step I (line 2), the TMRS is generated. In step II (line 4), an initial solution $\mathbf{S}_J$ is defined by the function $MakeInitSol$ which designates a random $\gamma_m \in \Gamma$, $\forall m \in \mathbf{M}$. In step III (line 5), a local search for $\mathbf{S}_{opt}$ is conducted to enhance the initial solution $\mathbf{S}_J$. The local search adapts an iterated greedy heuristics as described in Algorithm 5. Step II and III are repeated for $MaxIterations$ times, while the best solution is stored as $\mathbf{S}^*$ (line 6). The multi-start avoids the trapping in local optima during step II.

The local search step conduct an iterated greedy heuristics as explained in Algorithm 5. Given an initial streams set $\mathbf{S}_{init}$, the optimal nCGC is computed using

Algorithm 4: Multi-Start Iterated Greedy for DAMR

Input : G, M
Output: S*

1 $\mathbf{S}^* \leftarrow \emptyset$
2 Generate the TMRS: Γ
3 **for** *k = 1,2,..., MaxIterations* **do**
4 $\quad \mathbf{S}_J \leftarrow MakeInitSol(\Gamma)$
5 $\quad \mathbf{S}_{opt} \leftarrow LocalSearch(\mathbf{S}_J)$
6 $\quad \mathbf{S}^* \leftarrow UpdateSolution(\mathbf{S}^*, \mathbf{S}_{opt})$
7 **end**

the DASP technique in line 1. Once, the partition in graph $\mathcal{G}$ is obtained, the contribution of each message α_m, $m \in \mathbf{M}$ to the total CGC is determined according to (29).

$$\alpha\Big(m, J(m)\Big) = \sum_{i \in \gamma_m^{J(m)}, j \in \mathbf{S}/\{\gamma_m^{J(m)}\}} w(i,j) \tag{29}$$

The message of the most substantial contribution m^* is determined by:

$$m^* \leftarrow \arg\max_{m \in \mathbf{M}} \{\alpha\Big(m, J(m)\Big)\} \tag{30}$$

The DRS of m^* is replaced by the best DRS in Γ_{m^*} as in line 8. The message m^* is removed from *List* to ensure handling other messages. The optimization is terminated either when the optimized stream set is settled, i.e., $\mathbf{S}_{opt} = \mathbf{S}_{temp}$ after handling all messages, or when the maximum number of iterations $itrN$ is reached.

4.6 Iterated ILP-based Scheduling

Given the streams set $\mathbf{S}^*$, the IIS shall divide it into disjoint groups $S^1, S^2, ..., S^{N_G}$ and then search for a sub-schedule of one group $S^n, n = 1, 2, ..., N_G$ at a time. Every

Algorithm 5: Local Search

Input : $\mathbf{S}_{init}$, $\mathbf{M}$, N_G
Output: $\mathbf{S}_{opt}$

$(S^n, ncgc) \leftarrow GenerateDASP(\mathbf{S}_{init}, N_G)$
$\mathbf{S}_{temp} \leftarrow \mathbf{S}_{init}$
for $itrN$ **do**
  $List \leftarrow \mathbf{M}$
  while $List \neq \emptyset$ **do**
    Compute the message cost $\alpha(m, J(m))\ \forall m \in List$ according to (29)
    Find the message with largest contribution in the cost according to (30)
    $J(m^*) \leftarrow arg\min_{j \in [0, j_{max})}\{\alpha(m^*, j)\}$
    Remove m^* from $List$
    $(S^n, ncgc^*) \leftarrow GenerateDASP(\mathbf{S}_{init})$
  end
  $\mathbf{S}_{opt} \leftarrow \bigcup_{m \in \mathbf{M}} \gamma_m^{J(m)}$
  if $\mathbf{S}_{opt} = \mathbf{S}_{temp}$ **then**
    break
  else
    $\mathbf{S}_{temp} \leftarrow \mathbf{S}_{opt}$
  end
end

iteration takes the previous sub-schedules as additional constraints to prevent overlapped schedule. The sub-schedule for S^n is obtained by solving the ILP problem formulated as follows.

ILP Attributes – The proposed ILP formulation is defined by the following attributes:

- S^n: The streams group of interest.
- $P \equiv \{p_i | \, i \in S^n\}$: The periods of all streams in S^n.
- $L_i \subseteq E$: Set of links that compose the path of stream $i \in S^n$.
- $T \equiv \{t_{(i,l)} | \, i \in S^n, l \in L_i\}$: The transmission time of each frame in stream $i \in S^n$ over link $l \in L_i$.

- $C \equiv \{c|\, 0 \leq c < c_{max}\}$: The set of time slots. The hyper-period is descritized into C time slots. The number of time slots m is a common multiple of the number of frames per hyper-period of each stream. The duration of a time slot is denoted by T_c.

- B: An auxiliary big integer larger than c_{max}.

ILP Variables – The ILP formulation uses the following variables:

Transmission offset, $X \equiv \{x_{(i,c)} \in \mathbb{Z}_2|\, i \in S^n, c \in C\}$. $x_{(i,c)}$ is a binary variable indicates that the transmission of the first frame belong to stream i starts at the time slot $c \in C$ when $x_{(i,c)} = 1$. Otherwise, $x_{(i,c)} = 0$. The sender can transmit frames only at the beginning of the time slots. On the other hand, bridges do not adhere this rule, i.e., frames are forwarded once they are received and processed with no-wait.

Slots reservation, $H \equiv \{h_{(i,l,c)} \in \mathbb{Z}_2|\, i \in S^n, l \in L_i, c \in C\}$. $h_{(i,l,c)}$ is a binary variable indicates whether time slot c at link l is reserved for stream i when $h_{(i,l,c)} = 1$. Otherwise, $h_{(i,l,c)} = 0$. Each time slot can be designated for only one stream even if $t_{(i,l)} < T_c$. When $t_{(i,l)} > T_c$, multiple slots shall be reserved at l. In fact, smaller T_c allows denser schedule at the expense of problem complexity. It is important to note that the time slot reservation concerns the ILP solver and are not used to configure the egress ports in TSN bridges. Instead, the connected set of reserved slots on each link will be translated to a transmission window during which the TT gate is open.

Prior slot occupancy, $O \equiv \{o_{(l,c)} \in \mathbb{Z}_2|\, l \in E, c \in C\}$. $o_{(l,c)}$ is a binary variable indicates whether time slot c at link l is occupied by any stream in prior iterations when $o_{(l,c)} = 1$. Otherwise, $o_{(l,c)} = 0$.

ILP Constraints – The ILP constraints of IIS are as follows:

Transmission constraints: The first frame of each stream $i \in S^n$ has to be assigned to a time slot from the set $C_i \subseteq C$ where $C_i \equiv \{c|\, 0 \leq c < \lfloor \frac{p_i}{T_c} \rfloor\}$. This is

guaranteed by constraint (31).

$$\forall i \in S^n : \sum_{c \in C_i} x_{(i,c)} = 1 \tag{31}$$

This implies that stream i shall be assigned to an exact one time slot among the available ones $c \in C_i$. Limiting the search to C_i is to avoid the overlap between the frames in one hyper-period. The upper bound of the number of constraints in the shape of formula (31) is equal to $|S^n|$.

Time slots reservation constraint: The transmission of every frame in i implies reserving a set of time slots over the path links L_i. The set of Reserved Time-Slots (RTS) depends on the transmission offset and the required transmission time over each link $l \in L_i$. The RTS for stream i with transmission offset starts at time slot c is denoted by $\delta(i,c) \equiv \{(l,c') | \, l \in L_i,\, F_s(i,l,c) \leq c' \leq F_e(i,l,c)\}$, where (l,c') refers to reserving the link l at the time slot c'. $F_s(i,l,c)$ and $F_e(i,l,c)$ are computed by (32) and (33), respectively, and denote the time slots on which the link l starts and finishes the forwarding the first frame of stream i given $x_{(i,c)} = 1$.

$$F_s(i,l,c) = c + \left\lfloor \frac{1}{T_c} \cdot \sum_{l' \in L^*_{(i,l)}} t_{(i,l')} \right\rfloor - 1 \tag{32}$$

$$F_e(i,l,c) = c + \left\lceil \frac{1}{T_c} \cdot \left(t_{(i,l)} + \sum_{l' \in L^*_{(i,l)}} t_{(i,l')} \right) \right\rceil - 1 \tag{33}$$

where $L^*_{(i,l)} \subseteq L_i$ is the set of passed links before l. For example, let the stream i has the path $\mathcal{P}_i = [v_0, v_5, v_{10}, v_4]$ in the network shown in Fig. 4.2. Then, the corresponding RTS for i given $c = 0$ is $\delta(i,0) = \{(0,0),(0,1),(1,1),(1,2),(1,3),(2,3),(2,4)\}$ which is shown in Fig. 4.5. The size of $\delta(i,c)$ is constant for all $c \in C$ and denoted by

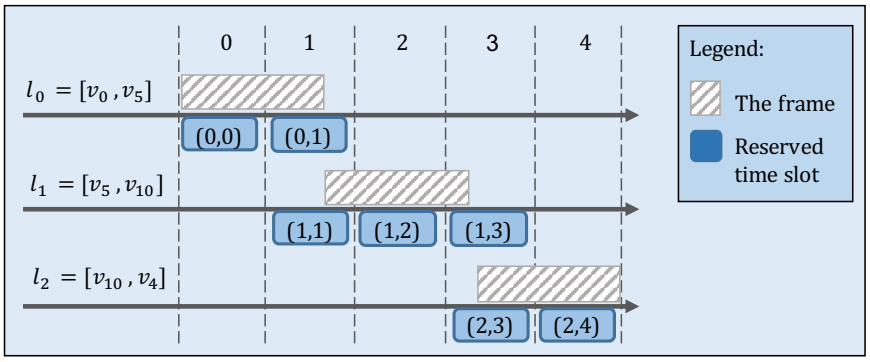


Figure 4.5: The set of reserved time slots for a frame routed over the path $[l_0, l_1, l_2]$ given an offset $c = 0$.

A_i which is seven in this example. In order to ensure that all time slots in $\delta(i, c)$ are reserved for stream i when $x_{(i,c)} = 1$, the constraint (34) is defined for every stream $i \in S^n$ and for every time slot $c \in C$.

$$\forall i \in S^n, \forall c \in C_i : B \cdot x_{(i,c)} + \left(A_i - \sum_{c' \in \delta_{(i,c)}} h_{(i,l,c')} \right) \leq B \tag{34}$$

The summation term checks the reservation of all needed time slots stream i to be able to transmit at the time slot c. This constraint enforces $x_{(i,c)}$ to be zero unless the value inside the brackets is zero. A_i is equal to the number of the variables $\{h_{(i,l,c')}, c' \in \delta(i, c)\}$ by definition, and thus the only way to have zero inside the brackets is by reserving the complete set $\delta(l, c)$. The upper bound of the number of constraints in the shape of formula (34) is equal to $|S^n| \cdot |C|$.

Frames isolation constraints: The schedule should maintain a temporal isolation between frames belong to different streams at every node in the network, i.e., the preceding frame leaves the queue before the succeeding one arrives. Such isolation eliminates the randomness of the queuing delay. Thus, we introduce the constraint (35).

$$\forall l \in E, c \in C : o_{(l,c)} + \sum_{i \in S^n} \sum_{c' = c \bmod p_i} h_{(i,l,c')} \leq 1 \tag{35}$$

This constraint states that each link l shall not be allocated to more than one stream at any point of time. First term $o_{(l,c)}$ prevents reserving the time slots which have been occupied in prior iterations. The summation term covers the streams in the current iteration S^n. Since each stream has a particular period p_i, the corresponding time slot of stream i is $c' = c \mod p_i$. The number of constraints in the shape of (35) is $|S^n| \cdot |C|$.

IIS Procedures – The procedures of IIS are presented in Algorithm 6. The inputs of IIS are the optimized streams set $\mathbf{S}^*$ and the number of groups N_G. When a valid schedule is found, the output X^* indicates the transmission offsets of each TT frame. Such offsets imply the open and close events for the gate of TT queue on each egress port. X^* can be translated into gate events by open the TT gate in agrees port i for the time slots c on which $x_{(i,c)} = 1$ and the gate shall be closed otherwise. In the initialization step (lines 1 to 3), the stream groups $S^0, S^1, ..., S^{N_G}$ are generated by the DASP technique, while O and X^* are set to zero. $\delta(i, c)$, $\forall i \in \mathbf{S}^*, \forall c \in C_i$ are defined according to (32) and (33). The schedule is built over N_G iterations. Each iteration handles a stream group S^n, $n = 1, 2, ..., N_G$ as described in lines 4 to 13. The corresponding ILP constraints of shapes (31), (34), and (35) are generated for S^n. The function $solvingILP()$ search for a feasible solution satisfies these constraints. The Boolean variable *done* is $True$ when a feasible solution is found. When a feasible solution is found for iteration n, O is updated in line 8 to preserve the occupied time slots for the next iterations, and X is added to X^* to save the sub-schedule as in line 9. The algorithm is terminated with unsolved schedule in two cases: (i) Timeout; (ii) The ILP solver found that solution of certain iteration is infeasible, i.e., $done = False$

as in line 11.

Algorithm 6: Iterated ILP-based Scheduling

Input : $\mathbf{S}^*$, N_G
Output: X^*

1 $S^n|_{n=0,1,...,N_G} \leftarrow GenerateDASP(\mathbf{S}^*)$
2 Initialize X^* and O
3 Obtain $\delta(i,c)$, $\forall i \in \mathbf{S}^*, \forall c \in C_i$
4 **for** $n \in N_G$ **do**
5 Generate the ILP constraints given S^n & O.
6 $\{X, H, done\} \leftarrow solveILP()$
7 **if** *done* **then**
8 Update O s.t $o_{(l,c)} = \sum_{i \in S^n} h_{(i,l,c)} \quad \forall l \in E, c \in C$
9 $X^* \leftarrow X^* + X$
10 **else**
11 **break**
12 **end**
13 **end**

4.7 Results and Discussion

An extensive analysis of wide range test cases is used to evaluate the DA/IRS. The evaluation is conducted in terms of the scalability and performance comparing to two recent techniques. Moreover, the impact of parameters selection on the efficiency, i.e., success rate is investigated. The reported results in this section have been carried out on a workstation with an Intel Core i7 6820HQ processor running at 3.0 GHz and 16 GB RAM. The three techniques are implemented on MATLAB R2017a.

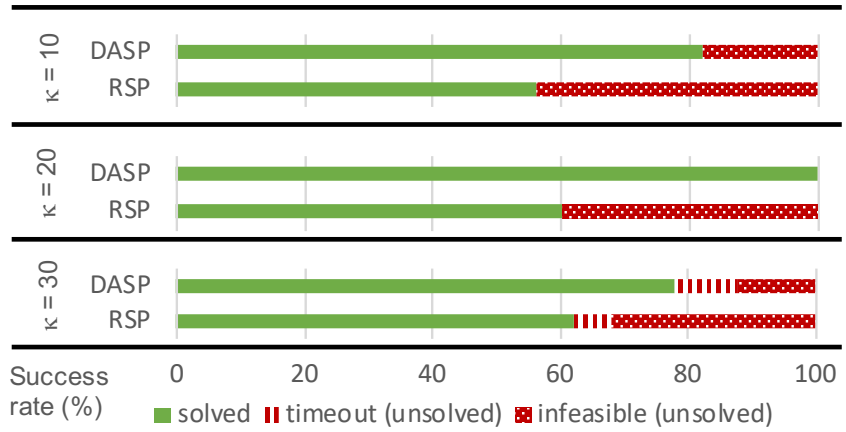
4.7.1 The Gain of DoC-Aware Streams Partitioning

DASP vs RSP: To demonstrate the advantage of applying the proposed DASP technique in the iterated scheduling, 60 test cases are analyzed. The test cases are based on a topology composed of eight ESs and eight bridges. The load is either 120

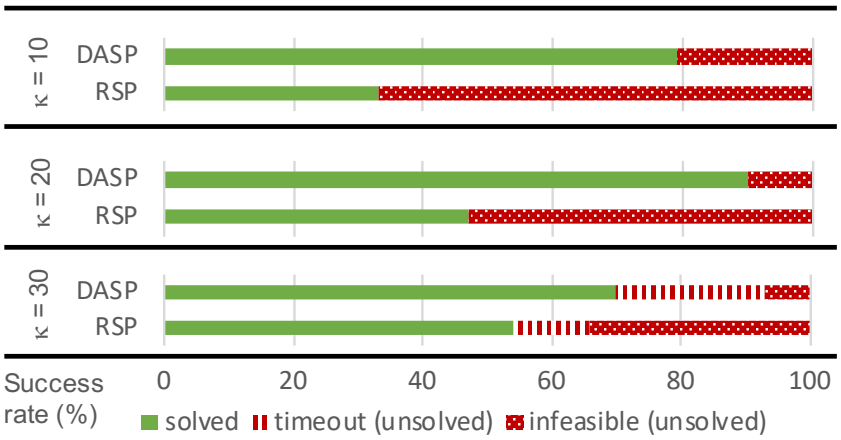
or 240 messages with RL equals 2, i.e., we have 240 or 480 streams, respectively. the source and destination of each stream are picked randomly from **ES**. The frames payloads vary between 50 and 1,500 Bytes, where streams periods vary between 5 ms and 100 ms. The streams are transmitted over 100 Mbps Ethernet links. Please note that the above streams setups and links bandwidth are applied for all experiments throughout this section. Each test case is iteratively scheduled using the IIS based on stream groups generated by both DASP and RSP. The required groups size is denoted by κ where $N_G = \lceil |\mathbf{S}|/\kappa \rceil$. We chose $\kappa = \{10, 20, 30\}$. The results shown in Fig. 4.6 which differentiates between the timeout unsolved instances and the infeasible ones. Please note that the time limit is 40 minutes, and the DAMR parameters are $Maxiterations = 3$, $j_{max} = 2$, $itrN = 10$, for all experiments.

The solid green portion of each bar in Fig. 4.6 represents success rate. It can be observed that DASP outperforms the RSP in all setups. The results show that timeout events start appearing in relatively large groups ($\kappa = 30$). Despite the higher overall success rate for the DASP, the RSP has less timeout instances. This can be explained by the fact that DASP increases the in-group conflict increases per-iteration runtime. On the other hand, RSP generates easier groups with lower in-group conflict. However, this is result in high CGC which prevents finding feasible solutions the latter iterations. All in all, the success rate of the highly utilized instances (480 streams, $\kappa = 20$) raised from 47% by the RSP to 90% by DA/IRS.

Impact of κ: The results in Fig. 4.6 show the trade-off between timeout and the solution feasibility. On the one hand, large groups help to handle all overlapped flows together which increases the chance to find feasible solutions, but this increases the likelihood of timeout termination as well. On the other hand, smaller groups do not suffer form the timeout termination, but in the expense of a limited schedule feasibility. The results in Fig. 4.6 suggest an optimal group size $\kappa = 20$ regardless the



(a)



(b)

Figure 4.6: The percentage of solved, timeout, and infeasible schedules of 60 synthetic test cases using DASP and RSP with different κ for (a) 240 streams; (b) 480 streams.

number of streams, and thus we will adopt this value in the next experiments. Please note that the optimum κ is based on the available resources in this experiments, and can vary for different memory and computational resources.

4.7.2 Performance Evaluation

The performance of the DA/IRS is evaluated in comparison with: (i) The ILP/JRS technique (Schweissguth et al., 2017); (ii) The PB/JRS teqhnique (Smirnov et al., 2017b). These techniques are adapted to support no-wait scheduling. The comparison is conducted in terms of runtime and the success rate. In the following, two experiments consisting of 140 synthetic test cases are presented.

Runtime vs traffic load: In this experiment we compare the runtime of the three techniques for different traffic load by varying the number of the stream, while fixing the network size. The number of streams ranges from 20 to 60 streams sent by 10 ECUs exchanging the streams over four bridges. Since ILP/JRS and PB/JRS do not support multipath routing, this experiment evaluates the time needed to generate a valid routing and scheduling for RL = 1. The runtime for 70 test cases averaged over the number of streams is shown in Fig. 4.7. The y-axis shows the average runtime of the successfully solved cases, i.e., the terminated cases due to the time limit do not contribute to the computed average. The graph demonstrates a 20x speed up by the DA/IRS comparing to PB/JRS and ILP/JRS. For example, PB/JRS technique takes around 1,600 sec on average to schedule 30 streams which is the maximum tackled size. Likewise, the ILP/JRS hardly tackles problems of 40 streams with an average runtime around 2,250 seconds. On the other hand, the proposed technique can solve problems of 60 streams within 100 seconds. Moreover, it can be noticed that the speed up gain is increasing with larger problems.

A comparison of the success rate for four traffic setups, 20, 30, 40, and 60 streams, is shown in Fig. 4.8. This figure shows that the DA/IRS has solved all 20-streams problems. On the other hand, PB/JRS technique has terminated without feasible solutions in almost 50% of those instances. For the highly utilized instances (60

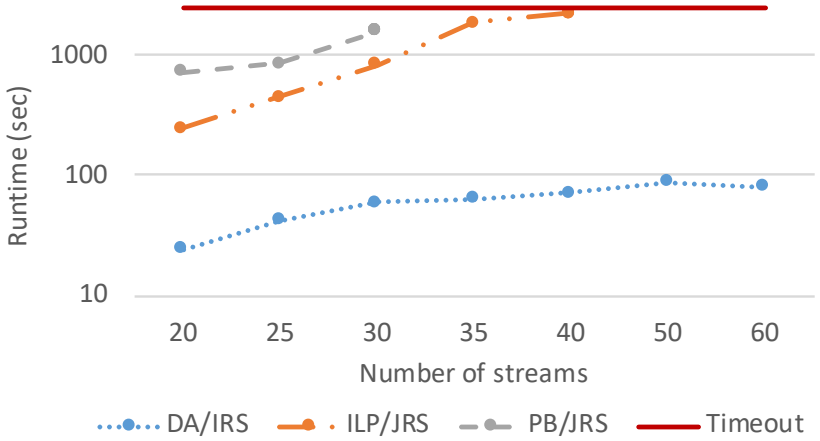


Figure 4.7: Average runtime in the log scale for the DA/IRS, ILP/JRS, and PB/JRS for 70 synthetic test with different number of streams.

streams), the DA/IRS was able to solve 90% of the test cases while other techniques have not finished. It can be noticed that the majority of unsolved cases by ILP/JRS and PB/JRS have been terminated due to the timeout constraint. This suggest that the prompt exploration for a limited search space achieves better success rate than the slow exploration of an extensive search space.

Runtime vs network size: In this experiment we compare the runtime of the three techniques for different network sizes. 70 synthetic test cases are handled with a number of bridges ranges from 3 to 21. The runtime results in shown in Fig. 4.9. Whereas, a comparison of the success rate of the three techniques for four network sizes of 3, 9, 15, and 21 bridges is shown in Fig. 4.10. It can be noticed from Fig. 4.9 that the runtime of PB/JRS and ILP/JRS shows a rapid increase with larger networks. For example, PB/JRS takes about 30 minutes to handle 6-bridges networks while it fails to determine a valid solution for any larger networks. Regarding the ILP/JRS, it shows slightly better scalability than PB/JRS. However, it still incapable of handling networks which have more than 9 bridges within the time limit as shown in Fig. 4.10. These results demonstrate the restrictions on the network size

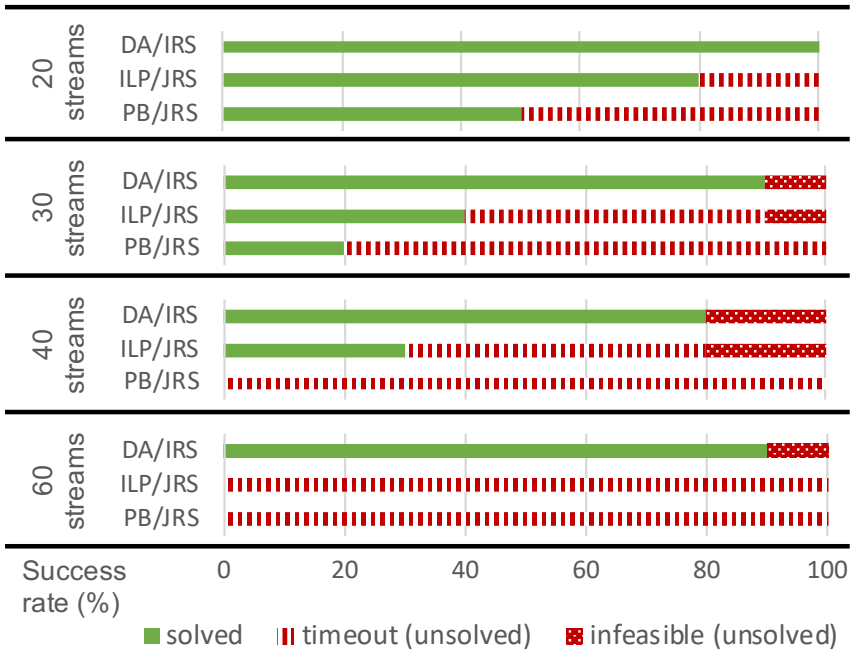


Figure 4.8: The percentage distribution of solved, timeout, and infeasible instances of the DA/IRS, ILP/JRS, and PB/JRS for different number of streams.

which can be practically handled by the existing techniques. Contrarily, the results demonstrate the well-scalability of the DA/IRS which shows an almost linear increase of runtime versus the network. On average the DA/IRS is up to 20-fold faster than other techniques. For example, it is capable of tackling 21-bridges networks in about 3 minutes. Moreover, the results in Fig. 4.10 show the superior success rate of the proposed technique over PB/JRS and ILP/JRS. Finally, we believe that the industrial requirements outpace the techniques capabilities and will likely continue to do so. The proposed DA/IRS takes a step towards closing such a gap by addressing the DoC problem to enable efficient incremental scheduling. However, standardized constraints formulation would enable interoperability tests, and allow better evaluation of different scheduling approaches, e.g., formal and heuristics. Moreover, open accessibility to the details of real world problems, which is still limited, is necessary to validate the theoretical solutions proposed in the literature and boost the future

TT traffic planning research.

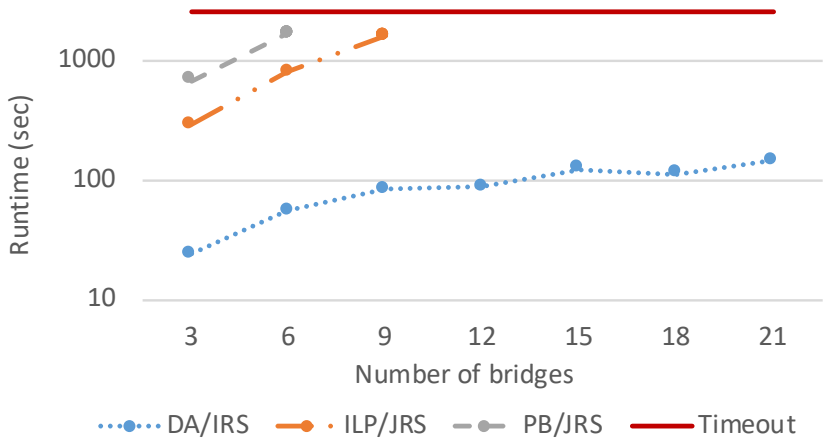


Figure 4.9: Average runtime in the log scale for the DA/IRS, ILP/JRS, and PB/JRS for 70 synthetic test cases of different number of bridges.

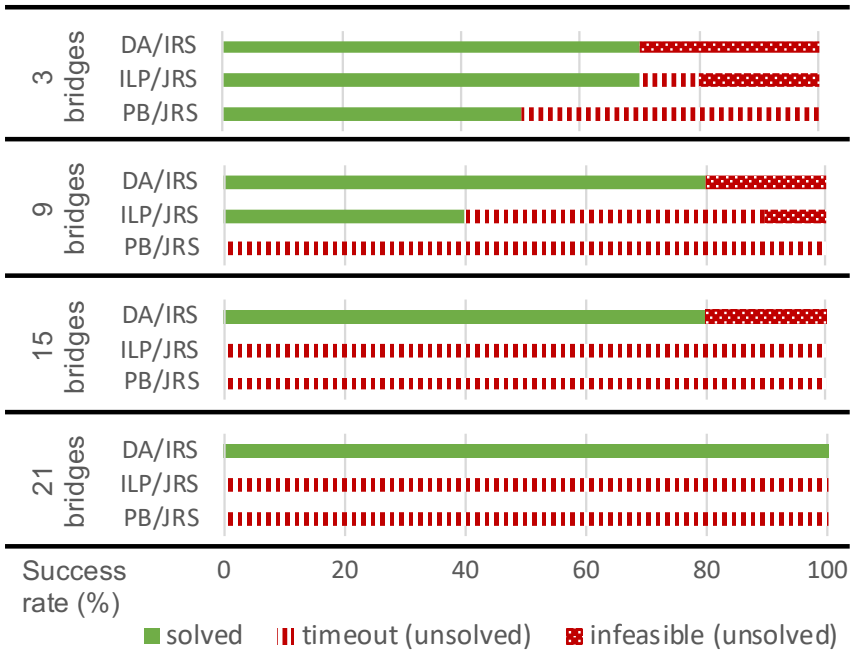


Figure 4.10: The percentage distribution of solved, timeout, and infeasible instances of the DA/IRS, ILP/JRS, and PB/JRS for different network sizes.

4.8 Conclusion

In this paper, a novel traffic planning method for TSN is introduced. The proposed DA/IRS method covers the multipath routing and the no-wait scheduling of TT traffic by the DAMR and IIS techniques, respectively. The DoC problem is formalized and addressed by the proposed DASP technique to improve the performance of the incremental scheduling. In particular, the DoC technique is proposed and integrated with the above techniques reduce the DoC and allow feasible schedules under high utilization conditions. Three experiments are conducted using 200 synthetic test cases to evaluate the scalability of the DA/IRS and its performance compared to two recent methods, namely, the ILP/JRS and PB/JRS. The results show the out-performance of the proposed method in handling large-scale as well as highly utilized networks. The DA/IRS is up to 20-fold faster than previous methods.

Chapter 5

Article II: Dynamic TSNs Priority Assignment for Fog Computing Using Reinforcement Learning

Authors: Ayman Atallah, Ghaith Bany Hamad, Otmane Ait Mohamed

Abstract: Fog computing refers to the intermediate nodes' shared computation capabilities, i.e., between Cloud and edge node to run time-critical applications. Such a paradigm requires reliable and very low-latency communication to ensure safe and stable operation. Time-Sensitive Networks (TSNs) are capable of delivering such strict requirements. TSN typically accommodates static traffic, and thus it defines a static traffic configuration, such as the flows' priority. The priority assignment is decided in the design phase to meet the flows' deadlines and usually optimizes the network in terms of performance and utilization. Such offline optimization is unsuitable for Fog computing, which generates dynamic traffic based on the executed tasks. Therefore, a dynamic priority assignment is essential to optimize the TSN

for Fog applications. This paper proposes a novel machine learning approach for dynamic flow priority assignments in TSN. In this approach, the priority of a new flow is assigned automatically by a Reinforcement Learning (RL) based agent. The agent is trained to optimize the flow timing behavior while maximizing the overall network throughput. For this goal, we formulate a cost minimization problem as a Markov Decision Process (MDP). The RL problem is solved by the Double Deep Q-Network (DDQN) technique to find the optimal priority assignment policy. We conduct extensive simulations adopting several synthetic test cases to evaluate the performance of the proposed approach. The results show higher admission rates and network utilization than a greedy heuristics technique.

5.1 Introduction

Industry 4.0 refers to the ongoing industrial revolution, which transforms every area in the industry thanks to the emerging computation and communication technologies. This revolution promises to boost productivity and flexibility, which reduce time-to-market and improve product quality. Realizing the concepts of Industry 4.0 entails a convergence of CPSs with state-of-the-art technologies (Pop et al., 2018). CPSs refer to networked computing elements connected to sensors and actuators to monitor and control physical processes. CPSs are typically deployed for safety-critical applications, such as healthcare, manufacturing, and transportation. Such applications require real-time control with a high level of reliability assurance beyond the Cloud computing paradigm (García-Valls, Cucinotta, & Lu, 2014). In this context, Fog computing paradigm is introduced to yield better CPSs integration for Industry 4.0 (Bonomi, Milito, Natarajan, & Zhu, 2014). Fog computing introduces a new system-level architecture that enables computing, storage, and control resources over

Fog nodes. These nodes operate as intermediate storage and computing elements deployed between the Cloud and the edge devices, e.g., industrial machines (Consortium et al., 2017).

Fog computing demands real-time communication due to the safety-critical nature of industrial applications. Furthermore, the complex architecture of the future industrial automation systems hikes the communication requirements in bandwidth and coverage (Finn, 2018). Conventional Ethernet technology offers scalable and cost-efficient connectivity. However, it is incapable of delivering real-time connectivity due to the non-deterministic queuing delay and packet collisions (Decotignie, 2005). IEEE 802.1 task group introduces new standards to realize low latency connectivity over scalable Ethernet architecture, namely TSN. TSN allows the coexistence of time-critical and best-effort traffic on the same network. This feature reduces the cost, reserves backward compatibility, and offers high flexibility to meet a broad spectrum of applications. The efficient implementation of Fog computing requires the efficient utilization of network resources. Meeting the deadlines of more flows implies a more significant portion of tasks executed by Fog nodes.

Fog computing generates dynamic traffic, which requires dynamic traffic planning and network reconfiguration to maintain high-performance (Haur & Chin, 2019). Such TSNs dynamic behavior is enabled in the extension IEEE 802.1Qcc where a Configuration Agent (CA) can accommodate and remove flows on the runtime (Gutiérrez et al., 2017). Nonetheless, optimized algorithms to compute such a dynamic behavior have been left out of the standard (Steiner et al., 2018). Several works attempt to address the problem over the last years, such as (Gutiérrez et al., 2017, 2015) which address the runtime scheduling, and (Pahlevan et al., 2019) which addresses the runtime routing. These works address the traffic routing and scheduling tasks. However, the dynamic PA has not been addressed yet. Static PA is incapable of capturing

the traffic patterns of each Fog node. Exploiting such patterns allows optimized PA decisions to increase the TSN's chance to admit more future flows. This paper is the first work that addresses the TSN's dynamic PA for Fog computing to the best of our knowledge.

This paper addresses the following research question: *how to perform a dynamic traffic reconfiguration that optimizes the TSN for Fog computing applications?* Efficient Fog computing implies the dynamic priority assignment of the new flow. TSN configuration should consider the transmission patterns generated by Fog nodes. Furthermore, the appropriate priority must meet the hard deadlines of the time-critical traffic while maximizing the network's long-term throughput. Solving such problems for static TSN is done at the design stage using exact methods, e.g., ILP, or meta-heuristic methods, e.g., Genetic Algorithms (GA). However, such methods are not suitable for dynamic traffic. In particular, the required processing time of such methods is too long, i.e., a couple of minutes, for Fog applications time scale. To address the above question, we explore the use of machine learning to obtain the optimal PA policy. Dynamic PA problem is formulated as a cost minimization problem using Markov Decision Process (MDP). To achieve the optimal flow priority assignment, we propose a Reinforcement Learning (RL) approach based on Deep Q-Network (DQN) (Volodymyr, Koray, David, Andrei, & Joel, 2015). The goal is to predict the best action (priority level) that maximizes the long-term reward. These rewards represent the number of accommodated flows while meeting their deadlines. An extensive simulation is conducted to evaluate the proposed approach's performance for different network setups. Our results show that our proposed approach delivers a higher admission rate than a typical greedy heuristics technique, i.e., RL/DPA gain the admission rate and network utilization up to 90% and 70%, respectively.

The remainder of the paper is organized as follows. Section 5.2 presents the related

works. The considered system model is described in Section 5.3. The proposed RL-based approach is introduced in Section 5.4. Section 5.5 reports the experimental results. Finally, Section 5.6 concludes the main findings of this work.

5.2 Related Work

The synthesis and analysis of static traffic in TSN is addressed in several works (Hamza, Scharbarg, & Fraboul, 2014; Maxim & Song, 2017; Smirnov, Glaß, Reimann, & Teich, 2017a). Authors in (Smirnov et al., 2017a) present a timing analysis of RC traffic considering the impact of the higher priority TT traffic. The impact of both TT traffic and the traffic shaping on AVB traffic timing is studied in (Maxim & Song, 2017). Whereas, Solving the PA problem for RC time-critical flows in switched networks is addressed in (Hamza et al., 2014). The adopted technique minimizes the worst-case end-to-end delay under two priority levels. Furthermore, the static priority synthesis for UBS architecture (Specht & Samii, 2016) is addressed in (Specht & Samii, 2017). An SMT based heuristics is adapted to solve the flow to queue and queue to the PA problem. Several approaches are investigated in (Imtiaz, Jasperneite, & Weber, 2012) to reduce the latency in the AVB traffic for industrial applications, such as interrupting or shortening the non-time-critical frames that can interfere with AVB frames.

Several works attempt to address the problem regarding TSN-based Fog Computing platforms and have pushed the performance boundaries over the last years. Dynamic configuration agents are introduced in (Gutiérrez et al., 2017, 2015) to allocate new TT flows on runtime. Such agents observe the transmission patterns of new flows to estimate their period, length, and latency, enabling them to generate schedules in the runtime. Moreover, the runtime scheduling of TT traffic in TSN is

addressed in (Raagaard et al., 2017) by a greedy heuristic technique for fast execution. This technique determines the GCL, which decides the transmission of TT frames on each egress port of a network switch. However, the literature mentioned above does not address the dynamic PA. Instead, the flows' priority is assumed to be fixed on the highest level. Such assumption does not apply on RC traffic which has a range of priorities.

The Machine Learning (ML) is applied in many aspects related to TSN. The authors in (Mai et al., 2019; Navet et al., 2019) investigate the speed up of the design space exploration in TSN using machine learning to replace the conventional schedulability analysis, e.g., network calculus. Supervised and unsupervised machine learning techniques are employed to test the schedule feasibility for a particular offline configuration. The RL is employed in (Y. Wang et al., 2018) for dynamic traffic and computation co-offloading in Fog computing implemented for mobile services in vehicular networks. The tradeoff between energy consumption and service delay is investigated by dynamic RL-based and deep RL-based scheduling techniques.

5.3 System Model

5.3.1 Architecture Model

Fog computing architecture is illustrated in Fig. 5.1. The network connecting Fog node to edge nodes is denoted as a Southbound connection. The links between Fog nodes and Cloud are denoted as northbound connections.

In this paper, we consider a TSN-based southbound connection in Fog computing platform, as shown in Fig. 5.1. The TSN comprises a set of Switchs (SWs) and a set of Edge Nodes (ENs) which includes multiple edge nodes and one Fog node.

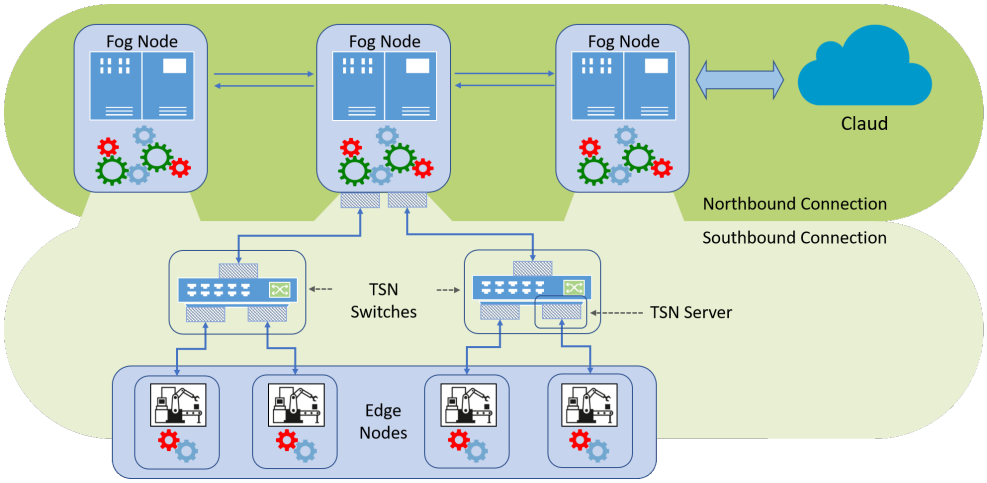


Figure 5.1: Fog Computing platform where Fog nodes located between the edge nodes and the Cloud.

Edge nodes can offload the computing workload to Fog node, equipped with more extensive storage and computing capabilities. Fog node is presumed to operate as a Central Network Configurator (CNC) to perform runtime network reconfiguration for the corresponding southbound connection. All network nodes $V = EN \cup SW$ are interconnected via full-duplex Ethernet physical links. The output port of each directional link is equipped with a TSN server. $\mathcal{S}$ denotes the set of servers in the network. A schematic of the considered TSN server is depicted in Fig. 5.2. The server $s \in \mathcal{S}$ contains a set of FIFO queues Q(s) where $|Q(s)| = K + 1$. The first K queues store Time-Critical (TC) flows. Whereas, the last queue stores not Time Critical (nTC) flows. Every queue $q \in Q(s)$ has a distinct priority level $0, 1, ..., K - 1$. TSN server selects the transmitted frame among the non-empty open queues according to a non-preemptive strict priority scheme. Selected frames are transmitted according to a constant transmission speed $r(s)$ of the connected physical link, e.g., 100MB/s or 1GB/s.

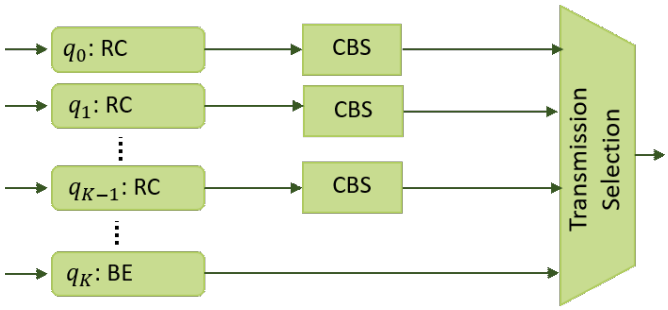


Figure 5.2: Egress port in TSN-compliant switch.

5.3.2 Application Model

A message is exchanged in TSN as a stream of frames transmitted from a specified source to one or more destinations with a specific path, size, and priority. TSN-compliant switches, namely bridges, store the forwarded messages in multiple FIFO buffers according to their priority. Three traffic classes are supported by TSNs namely, TT, RC, and BE classes. TT class provides deterministic connectivity, while the RC class guarantees bounded latency with way more scalability and flexibility. RC connectivity neither requires scheduled transmission nor clock synchronization. Instead, RC traffic follows a strict priority scheme integrated with credit-aware shapers to regulate the flow rate, which offers various delay bound for different flows.

The data is exchanged in TSN networks by the concept of flows. A flow f_i is a stream of frames transmitted from a particular source to one destination (unicast) or more (multicast) through specific routing. $\mathcal{F}$ denotes the set of all flows in the network. Whereas, *F(s)* denotes the set of flows pass over server *s*. Asynchronous communication is considered, i.e., frames have neither periodic patterns nor specific offsets. However, a flow must adhere to predefined transmission rate and burstiness. As shown in (36), the cumulative data $w_i(d)$ of flow f_i with a burstiness $\hat{b}_i$ over time interval d is limited to the upper bound $w_i(d)$.

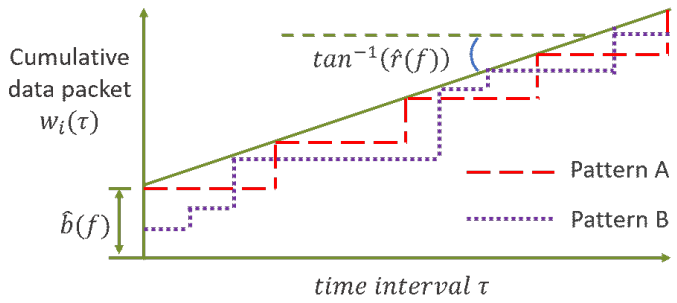


Figure 5.3: An example of two traffic patterns that satisfy the same leaky bucket constraint.

$$w_i(d) \leq \hat{b}_i + (\hat{r}_i \times d) \tag{36}$$

where $\hat{r}_i$ is the designated leaky bucket rate for flow f_i. The first K queues which store the TC traffic are headed by a CBS. The CBS is responsible for spreading the frames over time and regulate the traffic according to $\hat{b}_i$, and $\hat{r}_i$ is not exceeded. The CBS apply a frame-by-frame leaky bucket algorithm (Goyal & Vin, 1997; Zhang & Ferrari, 1993) to regulate the flows' transmission rate over the hops along their paths as shown in Fig. 5.3. This figure illustrates two different patterns (A & B) of flows that share the same parameter, i.e., burstiness and rate. Each flow f_i is assigned to a certain priority $p_i \in [0, K)$ which applies on all servers over its route where $\mathcal{P} = \{p_i | f_i \in \mathcal{F}\}$ denotes the flow PA in the whole network.

Each edge node $e \in \mathcal{E}$ runs a set of Fog applications. The application is run locally or offloaded by Fog node. The edge node generates offloading requests to the nearest Fog node. The offloading decision only depends on the network's capability to admit the corresponding flow while meeting all flow deadlines. The computing capacity of Fog node is assumed to be sufficient for all offloaded tasks. In this work, the offloaded task is represented as a flow fi, which is defined with burstiness bi, rate $\hat{ri}$, deadline $\bar{di}$, and expiry time te after which the flow is terminated due to

task completion. The new flow is either admitted or rejected by the CNC. Fog nodes offer way more computation resources than edge nodes. Therefore, a low admission rate wastes such resources and results in a lower performance of the system. In other words, the TSN ability to admit the time-critical flow is essential to utilize the Fog nodes efficiently. The CNC searches for a priority for fi such that all flows' deadlines are satisfied, including $\bar{d}i$. If a valid PA is found, the flow fi is admitted. On the other hand, if no valid solution is found, the flow is rejected, and the corresponding computing task is disposed locally on the edge node. The generate offloading request at each edge node follows a Poisson distribution with an arrival rate of λ_e. The transmitted flows from edge nodes to Fog nodes and vise versa are denoted as uplink and downlink connections. In this work, we consider Fog computing platform where the uplink is presumed the bottleneck of the offloading decision. In particular, the offloaded workload is assumed to require a large volume of input data over the uplink (Y. Wang et al., 2018). On the other hand, the downlink delivers a small volume of output results correspond to the offloaded workload. Thus, we consider the PA of unicast AVB flows, which are transmitted over the uplink to improve the computation offloading throughput.

5.3.3 Worst-Case Delay of RC Traffic

Every flow $f \in \mathcal{F}$ has a deadline $\bar{d}(f)$ specifies the acceptable time duration a frame can spend in the network before reach the sink $y \in Y(f)$. The application requirements define deadlines. On the other hand, the WCD of an RC flow f_i depends on the PA of f_i at each server $s \in R(f_i)$. In particular, the delay model is composed of $\hat{d}(f_i, sr_k, sr_{k+1})$ where $f_i \in \mathcal{F}, sr_k \in R(f)$ which describes the local worst case delay of f_i at each server $sr_k \in R(f_i)$ before reaching the next server sr_{k+1} (Specht & Samii,

2017). Consequently, the end-to-end delay bound $\hat{d}(f, y)$ to each sink $y \in Y(f)$ can be determined as (37).

$$\hat{d}(f, y) = \sum_{k=1}^{n-1} \hat{d}(f, sr_k, sr_{k+1}) + \hat{d}(f, sr_n) \tag{37}$$

where $\hat{d}(f, sr_n)$ is the delay bound from the last server sr_n to the destination. $\hat{d}(f, sr_k, sr_{k+1})$ is computed as the follows. Let $F_{HP}(f, s)$, $F_{SP}(f, s)$ and $F_{LP}(f, s)$ are the flows assigned to server s with a higher, same, and a lower priority level than $q(f, s)$, respectively. The delay bound $\hat{d}(f, s)$ is computed as Eq. (38).

$$\hat{d}(f, s, \mathcal{P}) = \frac{\hat{l}(f)}{r(s)} + \frac{\hat{l}_{LP}(f, s) + \sum_{g \in F_{HP}(f,s) \cup F_{SP}(f,s) - \{f\}} \hat{l}(g)}{r(s) - \sum_{g \in F_{HP}(f,s)} \hat{r}(g)} \tag{38}$$

Then

$$\hat{d}(f, s, s^+) = \max_{g \in F(q(f,s^+))} \left(\hat{d}(g, s) \right) \tag{39}$$

$$\hat{d}(q, s) = \max_{g \in F(q)} \left(\hat{d}(g, s) \right) \tag{40}$$

Given the timing constraints and the delay model of $\mathcal{F}$, the slack computed by (41) depends on $\mathcal{P}$.

$$slack(\mathcal{F}, \mathcal{P}) = \sum_{f \in \mathcal{F}} \max \left\{ 0, \hat{d}(f, y, \mathcal{P}) - \bar{d}(f, y) \right\} \tag{41}$$

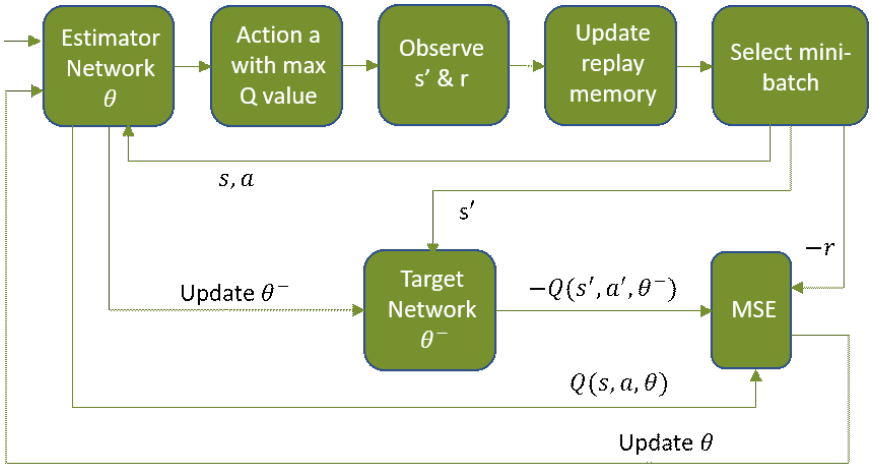


Figure 5.4: The architecture of the reinforcement learning model in our system.

5.4 Proposed Reinforcement Learning-Based Priority Assignment

5.4.1 RL/DPA procedures

The architecture of the RL/DPA is illustrated in Fig. 5.4. On each time step, the current state is passed to the estimator network to estimate each possible action's Q value. One action is selected based on the $\epsilon - greedy$ policy and passed to the environment. The experience tuple of the current state, action, reward, and next state $\langle s, a, r, s' \rangle$ is pushed into the replay memory until it is populated with a predefined size of samples. For every iteration i, a mini-batch is chosen based on a prioritized criteria to train the estimator network to enforce $Q(s, a, \theta^*) \simeq Q^*(s, a)$. The current estimator network's parameters θ_i are updated to minimize the MSE represented by $L_i(\theta_i)$. The Q value $Q(s, a, \theta_i)$ and the target value Y_i^{DDQN} are calculated by two separate networks to increase stability and reduce the over-estimation. Once the estimator network is well-trained, i.e., $Q(s, a, \theta^*) \simeq Q^*(s, a)$, the optimal policy π^* can be applied by selecting the action with the max Q value.

The procedure of the proposed RL/DPA is shown in Algorithm 7. The goal is to equip the CNC with an RL agent that applies the optimal policy for the dynamic PA of new AVB flows. The policy's objective is to maximize the total number of accommodated flows while satisfying their hard deadlines. The RL agent first constructs an estimator and target networks with random parameters θ and θ^{-}, respectively. Then, the agent runs some trajectories to populate the experience replay memory. A mini-batch is randomly sampled from the replay memory to update *theta*. On every training iteration, memory elements get new priorities, which defines their sampling probabilities. The agent chooses actions based on the $\epsilon - greedy$ policy. The agent eventually learns the optimal PA policy.

5.4.2 Reinforcement Learning Formulation

Our goal is to find the optimal policy for CNC to maximize the total number of accommodated flows while satisfying their hard deadlines. Such a problem is formulated as a MDP comprises state space, action space, and reward.

State space

At each time t, a new flow arises f_t, the agent observes the traffic in the network and the parameters of f_t. These inputs are denoted as the current state $s \in \mathbf{S}$, where $\mathbf{S}$ is the complete state space. The state s expresses the traffic volume at each queue represented by the maximum delay at the queue $D \equiv \{d(s,q) | \, g \in G, q \in Q(g)\}$ based on the flows assigned to the queue as computed in (40). Additionally, state s describes the routing $\mathcal{R}_t$, burstiness, rate, and deadline of the new flow f_t. The state space is represented as follows.

Algorithm 7: DQN-based dynamic priority assignment

Input : Discount factor γ, exploration rate ϵ, initial replay memory m_{init}, number of steps N_{step}

Output: Function approximator $\mathcal{Q}(s, a, \theta^*)$

Initialize an empty experience replay memory Ω
Initialize the reward approximator $\mathcal{Q}$ with random weights θ_0 & target network $\hat{\mathcal{Q}}$ with $\hat{\theta}_0 \leftarrow \theta_0$
repeat
    Reset the environment with a randomly initial state s_0
    while $s' \neq s_{terminate}$ **do**
        Select a random action a_t
        Pass a_t to the environment, and observe reward r_t and next state s'
        Push the sample (s_t, a_t, r_t, s_{t+1}) to Ω
    end
until *Populate* Ω *with* m_{init} *samples*;
$i \leftarrow 0$
repeat
    Reset the environment with a randomly initial state s_0
    while $s' \neq s_{terminate}$ **do**
        Generate a random number ζ between 0 and 1
        if $\zeta \leq \epsilon$ **then**
            Select a random action $a_t \in \mathbf{A}$
        else
            $a_t \leftarrow \arg\max_{\bar{a} \in \mathbf{A}} \mathcal{Q}(s_t, \bar{a}, \theta_i)$
        end
        $\epsilon \leftarrow \epsilon - \Delta\epsilon$
        Pass a_t to the environment, and observe reward r_t and next state s'
        Push the sample (s_t, a_t, r_t, s_{t+1}) to Ω
        Pop a mini-batch $Omega_{mini}$ of K samples from Ω based on the prioritized criteria in (55)
        Define the function Y_i and $L_i(\theta_i)$ based on (51) and (50)
        $\theta_{i+1} \leftarrow$ Train $\mathcal{Q}$ to minimize the loss function $L_i(\theta_i)$
        $i \leftarrow i + 1$
    end
    if $i \mod N_{update}$ **then**
        Update the target network $\hat{\mathcal{Q}}$: $\hat{\theta} \leftarrow \theta_i$
    end
until $i = N_{step}$;

$$s = \begin{bmatrix} \mathcal{R}_t & \hat{b}_t & \hat{r}_t & \bar{d}_t \\ d(sr_1, q_1) & \dots & \dots & d(sr_1, q_K) \\ \vdots & \ddots & & \\ d(sr_M, q_1) & \dots & \dots & d(sr_M, q_K) \end{bmatrix} \tag{42}$$

Action space

Based on the current state s and the PA policy, the CNC decides an action, i.e., assigns a priority level for f_t or decide to dismiss it. Hence, the action space in our Markov Decision Process (MDP) is denoted as follows.

$$a = \left[a^d, a^1, a^2, \dots, a^K \right] \tag{43}$$

where $a^k \in \{0, 1\} | k \in [1, K]$ means that f_n designates the priority k if $a^k = 1$. Selecting the action $a^d = 1$ means that the network shall not accommodate the flow f_t, and the corresponding workload shall be computed locally on the edge node. Note that f_t shall has a distinct priority level, thus $\sum_{k \in A} a^k = 1$.

Reward

We aim to maximize the number of accommodated flows before violating a deadline, i.e., $slack(\mathcal{F}, \mathcal{P}) = 0$. Therefore, the agent is rewarded for every time step t on which a new flow is successfully added as follows.

$$r_t = \begin{cases} 1 & , slack(\mathcal{F}, \mathcal{P}) = 0 \\ 0 & , Otherwise \end{cases} \tag{44}$$

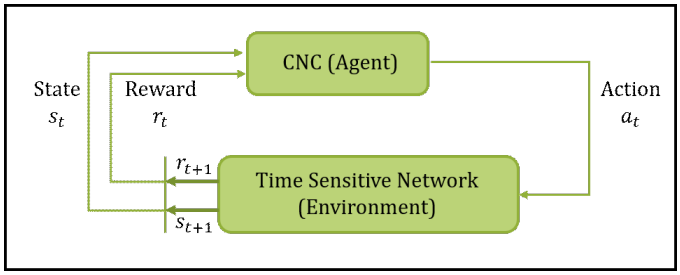


Figure 5.5: Reinforcement learning model.

5.4.3 Deep Double Q-Learning

The RL framework which is shown in Fig. 5.5. An agent repeatedly interacts with an environment for some steps $t = 0, 1, ..., T$ forming one trajectory. The agent observes the current state of the environment $s_t \in \mathbf{S}$, where $\mathbf{S}$ is the complete state space. Then the agent selects an action $a_t \in \mathbf{A}$ where $\mathbf{A}$ is the complete action space based on certain policy π which is a mapping form every state $s \in \mathbf{S}$ to a specific action $a \in \mathbf{A}$, i.e., $\pi : S \rightarrow A$. Afterward, the agent receives the new state s' (s_{t+1}) and the reward r_t which indicates the quality of selecting action a_t under state s_t. By training over many trajectories, the agent learns the optimal policy π^* to maximize the cumulative reward, which is defined as follows.

Definition 6. *For a trajectory i, the cumulative reward R_i is a weighted sum of the rewards resulted from all steps comprises trajectory i as computed by (45):*

$$R_i = \sum_{t=0}^{T} \gamma^t r_t \tag{45}$$

where the discount factor, $0 < \gamma \leq 1$ controls the valuation of earlier rewards over those received later.

Our case aims to maximize the number of accommodated flows assuming a direct

effect on offloading ratio. However, the proposed method opens the door to optimizing other Fog computing performance measures, such as network utilization and throughput.

We can define a policy that decides action to be taken in each state. Given the randomness in the added and removed flows, the optimal policy shall yield the maximum expectation of the accumulative reward as follows.

$$\pi^* = \arg\max_{\pi} E\left[\sum_{t\geq 0} \gamma^t r_t | \pi\right] \tag{46}$$

where E[x] is the expectation of x. For a policy π, the Q-value function for state s and action a provides the expected cumulative reward from taking action a in the state s as follows:

$$Q^{\pi}(s,a) = E\left[\sum_{t\geq 0} \gamma^t r_t |\, s_0 = s, a_0 = a, \pi\right] \tag{47}$$

Then the optimal Q-value function $Q^*(s,a)$ is the maximum cumulative reward that can be attained starting from the state s taking action a as follows:

$$Q^*(s,a) = \max_{\pi} E\left[\sum_{t\geq 0} \gamma^t r_t |\, s_0 = s, a_0 = a, \pi\right] \tag{48}$$

Then, $Q^*(s,a)$ satisfies the Bellman equation (49):

$$Q^*(s_i,a_i) = E\left[r_i + \gamma \max_{a_{i+1}\in \mathbf{A}} Q^*(s_{i+1},a_{i+1}) | s_i, a_i\right] \tag{49}$$

To estimate the optimal Q-value for each (state, action) pair, we use a function estimator $Q(s, a, \theta)$ based on a deep neural network with parameters θ. This estimator is known as Deep Q-Network (DQN) (Volodymyr et al., 2015), which is trained to enforce the Bellman equation, such that $Q(s, a, \theta^*) \simeq Q^*(s, a)$ where θ^* are the optimal parameters. The parameters *theta* are updated by back-propagation to minimize the Mean Square Error (MSE) of the following loss function (50).

$$L_i(\theta_i) = E\left[\left(Y_i - Q(s, a, \theta_i)\right)^2\right] \tag{50}$$

Where Y_i is the target value.

$$Y_i = r + \gamma \max_{a' \in \mathbf{A}} Q(s', a'; \theta_i) \tag{51}$$

Employing two networks, namely Double DQN (DDQN), is introduced in (Van Hasselt, Guez, & Silver, 2016) to reduce the over-estimation and improve the learning stability. The second network, namely target network $Q_{target}(s, a, \theta^-)$ can be employed to compute the DDQN target value Y_i^{DDQN}. In particular, Y_i^{DDQN} refers to the estimator network to decide the best action but uses the target network to compute its Q value as in (52).

$$Y_i^{DDQN} = r + \gamma Q_{target}(s', \arg\max_{a'}(Q(s', a'; \theta)), \theta^-) \tag{52}$$

The target network architecture is identical to the estimator network, while its parameters θ^- are updated every certain number of iterations based on θ by (53).

$$\theta^- = \alpha\theta + (1 - \alpha)\theta^- \tag{53}$$

where $\alpha \in [0, 1]$ indicates the update rate with $\alpha = 1$ is the highest rate in which the

estimator network is copied to the target network.

5.4.4 Prioritized Experience Replay and $\epsilon - greedy$ policy

At every step during trajectories, the observed tuple of state, action, reward, and next state $\langle s, a, r, s' \rangle$ is stored in the experience replay memory. Every iteration, a mini-batch of samples is chosen from the replay memory to update the estimator network. A prioritized sampling for the mini-batch is adapted to attain faster and better learning (Schaul, Quan, Antonoglou, & Silver, 2015). Such an approach tends to choose the samples that cause higher temporal difference error δ_i computed by (54).

$$\delta_i = |Q(s, a; \theta)_i - Q_{target}(s, a)_i| \tag{54}$$

Every sample i in the replay memory is ranked based on δ, i.e., the sample's rank with the highest temporal error equals 1. Afterward, the priority p_i of a sample i is inversely proportional to its rank. The probability P_i of choosing i is defined by (55).

$$P_i = \frac{p_i^\tau}{\sum_k p_k^\tau} \tag{55}$$

where $\tau \in [0, 1]$ controls the prioritization, i.e., $\tau = 0$ cuts out the prioritization.

To ensure different search space parts are sufficiently explored, we adopt the $\epsilon - greedy$ policy in choosing actions. With probability ϵ, the agent selects a random action (exploration). Otherwise, select greedy action based on the current estimator network (exploitation). The ϵ starts from a substantial value and then linearly decreases gradually as the iterations number increases.

5.4.5 Tentative Actions

After obtaining the Q value for every priority level for the new flow at a particular step, invalid actions that violate one or more deadlines at the network must be masked. The masking is done by adding large negative values to their corresponding Q values. The combination of the Q values and tentative action penalties are considered to decide the selected action.

5.5 Performance Evaluation

5.5.1 Experiment Settings and Metrics

The proposed RL/DPA performance is evaluated by the following two metrics: the admission and average network utilization. The admission rate is computed by dividing the number of admitted flows by the total number of offloading requests during one episode of one-hour duration. The average utilization is computed by (56).

$$U = \frac{1}{BW \times T_{episode}} \times \sum_{i \in I} \hat{r}_i \times T_i \tag{56}$$

where I is the total number of admitted flows, and T_i is the active time of flow $i \in I$.

5.5.2 Quantitative Results

The performance of the RL/DPA is evaluated in comparison with the greedy PA technique, based on the Optimal PA (OPA) (Hamza et al., 2014). The OPA technique adopts an offline greedy search. The adapted on-line search version used in this work for comparison is illustrated in Fig. 5.7. In the following, two experiments consisting

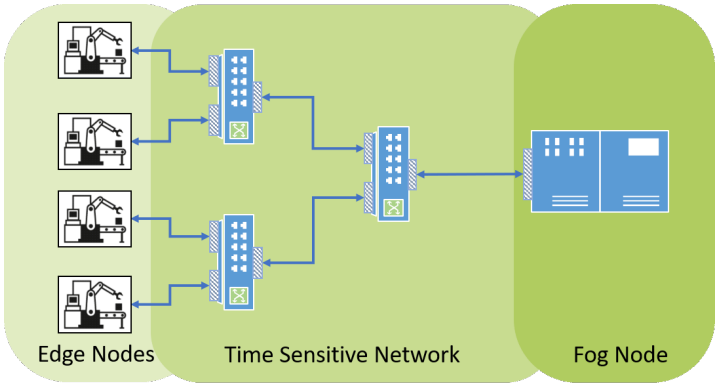


Figure 5.6: TSN topology connects four edge nodes to Fog node.

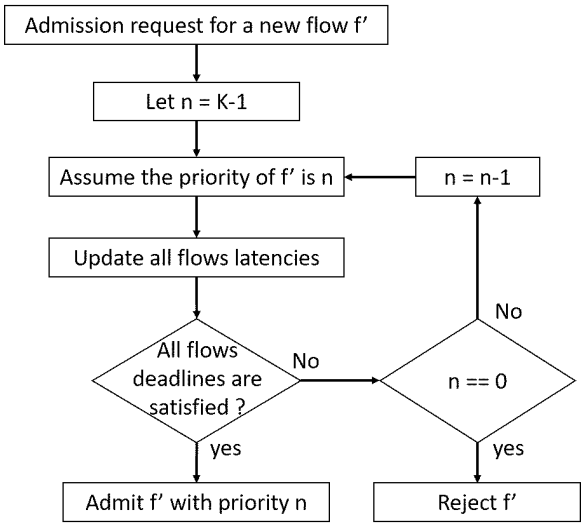


Figure 5.7: Flowchart of the greedy priority assignment technique.

of 23 synthetic test cases are presented. This work's reported results have been carried out on a workstation with an Intel Core i5-8500 processor running at 3.0 GHz and 16 GB RAM. The development environment is built using Python on the top of the Tensorflow framework. The parameters of the DQN are shown in Table 5.1

admission rate: In the first experiment, we use 20 test cases with either 100 or 200 applications. The network topology is shown in Fig. 5.6 is used for all test cases throughout this section. The physical link bandwidth ranges from 100 Mbps to 1 Gbps interconnect 10 ECUs exchanging the three bridges' flows. The admission rate

Table 5.1: Please write your table caption here

Parameter	Value
Initial replay memory size	5000
Maximum replay memory size	50000
Minibatch size	128
Target network update rate	2000
Discount factor	0.99
Learning rate	0.00001

attained by the RL/DPA, and the Greedy Priority Assignment (GPA) techniques for 200 applications is shown in Fig. 5.8 under different link bandwidth ranges from 100 Mbps to 1 Gbps. The graph demonstrates a considerable improvement of the TSN ability to serve Fog applications using the proposed RL/DPA. In particular, RL/DPA allows the TSN to admit up to 90% more flows comparing to GPA. Such gains allow the TSN to meet specific performance requirements using lower bandwidth links. For example, the RL/DPA allowed the TSN to admit 60% of the offloading requests using 400 Mbps links. On the other hand, the TSN needs a 1 Gbps link to attain the same admission rate by the GPA technique under the same traffic conditions.

The admission gain attained by RL/DPA compared to GPA for both 100 and 200 application test cases is shown in Fig. 5.9. It can be noticed that the gain of the admitted requests increases by increasing the traffic load (e.g., 200-application cases). The graph shows again drop after a specific bandwidth. This drop is due to the saturation of the RL/DPA admission rate is approaching 100%.

Network utilization: In the second experiment, we show the average network utilization of five test cases with a different number of applications, as shown in Fig. 5.10. The number of Fog applications ranges from 50 to 250 applications, while the links' bandwidth is 100 Mbps. The results show a better utilizing of the network proportional to the traffic load. For instance, the two approaches closely utilize the

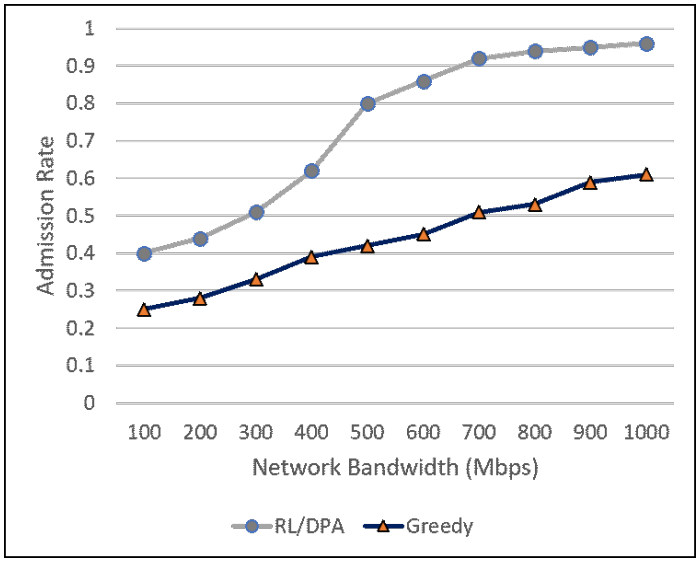


Figure 5.8: The admission rate achieved by the RL/DPA, and the greedy techniques for 200 applications under different link bandwidth.

Figure 5.9: The admission gain achieved by the RL/DPA, over the greedy technique for 100 and 200 applications.

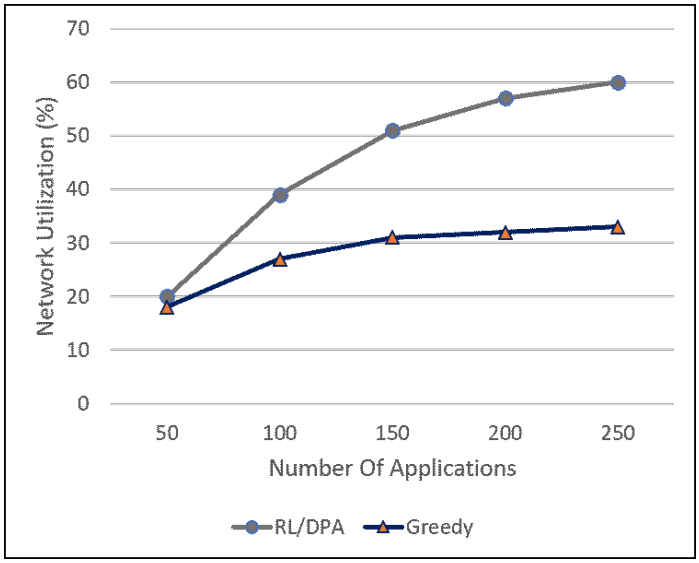


Figure 5.10: The average network utilization attained by the RL/DPA, and the GPA techniques for different number of applications.

network in the case 50 application, where the RL/DPA doubles the network utilization for the 250-application cases.

5.6 Conclusion

In this work, we solve the PA problem for dynamic traffic in TSN using a deep reinforcement learning model, namely RL/DPA. The dynamic traffic is centrally controlled by CNC that decides to admit a new flow with an assigned priority or reject it. To handle the dynamic traffic model for Fog applications, we propose a double deep Q network (DDQN) with prioritized experience replay. The model is trained to learn an optimal policy to maximize the admission rate as w ell as the network utilization. Results demonstrate that the proposed RL/DPA allows the optimized TSN to serve more Fog applications with lower link capacity. The RL/DPA outperforms the greedy PA in both admission rate and network utilization up to 90% and 70%, respectively.

Chapter 6

TSN Verification Under Single Event Upsets

6.1 Network Reliability Analysis Under Faulty Switch

TSN must maintain a synchronized TT transmission after one link is down. In other words, the failure in one link is isolated from the rest of the network. This assumption cannot hold in case of failures where the messages are still sent but at the wrong time. However, this assumption is not always valid. For instance, corruption in the configuration memory in one switch can affect the TT transmission in the whole network. This section describes the considered fault model and the proposed analysis of the impact of the faults on communication reliability.

6.1.1 Fault model under consideration

Each egress queue is connected to a gate that operates according to the transmission schedule stored in the GCL. Every row in the GCL, namely, Gate Control Entry (GCE), controls the status of the gates for a specific time window of width T_i. We

assume that each gate's state at each GCE is physically represented by one bit where the values of '1' and '0' refer to open and close state, respectively. After reaching the last entry GCE_N in the list, the TAS repeats the GCL for the next cycle. The column in the red box corresponds to the GCE of the critical traffic. Thus, we call this vector the critical gate control vector (CGCV). In the early design stage, the exact mapping between memory locations that store the GCL and the behavior of the timed gates is not available. Therefore, we propose a high-level abstraction for the bit-flip faults in GCL. In particular, a fault in GCL is abstracted and represented as a random change in the corresponding timed gate's behavior for a certain GCE in terms of status (opened or closed) and the duration.

6.1.2 TSN Reliability Analysis Using Model Checking

The TSN network and the fault behavior are modeled as Priced Timed Automata (PTA) based on the multi-agent architecture. According to the network model, the ES agent comprises three units, as illustrated in Fig. 6.1: the application unit automaton, the transmitter unit automaton, and the receiver unit automaton. Each switch is separated into several agents to represent the egress ports. A single egress port agent is composed of three units, as shown in Fig. 6.1: the TAS unit automaton, the Transmission Selection (TS) unit automaton, and the buffer unit automaton. The communication direction of frames between units is shown as solid arrows. The direction of control and observation between units is depicted in dashed arrows. The transmitter unit of the source ES sends a frame every period while the receiver unit of the destination ES notifies the application unit of delivered frames. The TAS is responsible for applying the GCL while the TS unit transmits the buffered frames

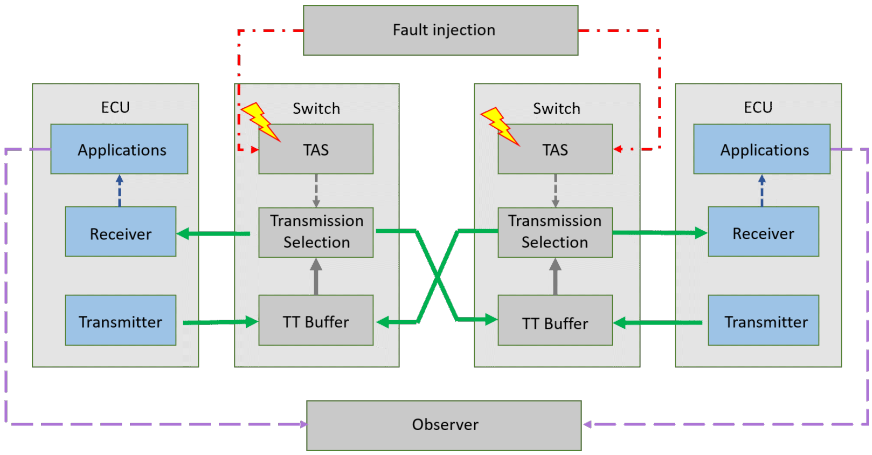


Figure 6.1: Proposed architecture of the networked PTA model of TT TSN.

according to their next-hop while the gate is open. The buffer pops the frame according to the FIFO scheme. On every deadline, the application unit checks if one replica has been received, at least. Otherwise, a *communication failure* is detected.

This modeling is implemented on UPPAAL SMC. UPPAAl is "a toolbox for modeling, simulating, and verifying real-time systems based on constraint-solving and on-the-fly techniques" (Papadopoulos, 2004). It represents the systems as a timed automata network, extended with integer variables, structured data types, and channel synchronization (Margaria & Steffen, 2010). UPPAAL is widely adapted for real-time controllers and communication protocols in which the representation of time is essential. UPPAAL can explore the state-space of the design, and preform reachability analysis to check invariant and reachability properties. One of the powerful features of this toolbox is "the application of on-the-fly searching techniques combined with the symbolic technique that reduces verification problems to that of manipulating and solving simple constraints" (Larsen, Pettersson, & Yi, 1997). In the case of unverified properties, a counterexample can be generated to explain how the property is violated. Furthermore, the generated counterexamples are graphically visualized

using the toolbox (Larsen et al., 1997).

The proposed PTA model is utilized to present the specific implementation of each traffic setup. Consequently, the fault-tolerance properties are verified for each setup. For each traffic setup, the GCL for each egress port in the network is generated. After that, extensive fault injection is performed on each GCE to verify the network reliability. In each fault injection scenario, the application models in all ESs are checked to identify whether the `failure` state is reachable. Moreover, the application units count the number of consecutive failures ψ and the number of consecutive deliveries τ. Hence, we characterize the worst-case failure pattern by examining all possible fault scenarios in GCLs. In particular, we specify the minimal control reliable interval τ_{min} and the maximal failure interval ψ_{max}, which refers to τ and ψ in the worst-case fault scenario.

6.2 Availability of SRAM-FPGA For TSN Switches

The flexibility and short design cycle of Field-Programmable Gate Arrayss (FPGAs) make them suitable to implement TSN-compliant switches, known as bridges (Li et al., 2018). Moreover, adapting FPGA-based components in high-energy radiation environments has attracted increasing attention due to their low cost and reconfigurability (Caffrey et al., 2002). In particular, Static Random-Access Memory (SRAM)–based FPGAs are widely employed for satellite data processing systems (Norton, Werne, Pingree, & Geier, 2009; Siegle, Vladimirova, Ilstad, & Emam, 2016). SRAM FPGAs have already been deployed in the Orion CEV interplanetary spacecraft (TTTech, 2014). However, the configurations that define the programmable logic units' functionality are susceptible to corruption when exposed to high-energy

radiation, namely Single Event Upsets (SEUs). Consequently, the transmission reliability of the TSN may be affected. Different vendors provide radiation-hardened FPGAs to withstand harsh radiation environments (Swift et al., 2011). Those FPGAs are expensive and lagging their non-radiation hardened FPGA counterparts. Therefore, mitigation techniques, such as periodic repair, are introduced to allow using commercial SRAM FPGA in the harsh space radiation environment (Siegle et al., 2016). Nonetheless, SRAM FPGA-based networks may suffer from downtime in the presence of SEUs and during the repair process. In short, space applications must investigate the availability of non-radiation hardened SRAM-based FPGA.

6.2.1 Timed Automata Model of TSN

In this section, we introduce a Timed Automata (TA) model representing the network topology, the routing and scheduling of critical messages, and the failure rate. Then, the network's availability is investigated under various repair times and rates using Statistical Model Checking (SMC) approach. The proposed analysis allows the designer to search for the proper repair mechanism required to meet network availability requirements. Furthermore, the tolerated failure rates of the individual switches can be estimated for given availability requirements. This analysis is especially crucial for TSN to reduce the classical redundancy overheads in such critical systems.

The proposed model tracks the TSN's availability concerning a particular higher-level application that involves a set of data streams from different sources. The TSN application is represented as a stream of messages transmitted from a source end-system and received by a specific destination end-system. The application `id` is modeled by one TA as shown in Fig. 6.2(a). In every cycle, the application sends

a message at certain offset `Offset[id]` to the first switch in the routing, stored in `Src[id]`. Hence, the automaton moves from state `Send` to state `TSN`. Then, the TA transits from state `TSN` according to the synchronization channel `send[id]?`, either to state `Routing` or `Receive`. If the current switch (`Pid`) is not the last node before the destination which is stored in `Dest[id]`, the automaton moves to state `Routing`. In that case, the next node in the routing is `R[id][hops]`, where `hops` indicates the number of nodes that have been crossed. If the waiting time in TSN exceeded the deadline of the message, the automaton moves to state `Error` and notifies the observer TA of the missed message via the synchronization channel `miss[id]!`. The application sends a new message after the time interval `Period`, which is equal to the cycle duration.

The TA model of the availability observer is shown in Fig. 6.2(e). At any time instant, the automaton can either be in state `Available` or state `Unavailable`. The transitions between these two states are enabled by the synchronization channels `miss[id]?` and `hit[id]?`. These channels are triggered by the application TA when a message instance misses or reaches the destination, respectively. The TA of the output port `Pid` in the TSN switch is shown in Fig. 6.2(b). The automaton moves from state `Available` to state `Receive` by the synchronization channel `arrival[Pid]` which is triggered by the stream TA. The message waits for a time interval `t<=Qmax`, where `Qmax` refers to the WCD at the switch. The TA of the fault injection module is shown in Fig. 6.2(c). The proposed model assumes that the switch goes out of service in two cases: i) SEU in the FPGA configuration bits, which lasts until the next repair; ii) periodic repair mechanism, i.e., scrubbing that lasts for `repair_time` (shown in Fig. 6.2(d)). In the fault-injection TA, the automaton leaves state `Normal` according to the exponential rate `lambda` which represents the failure rate in the configuration memory, i.e., the unit of λ is $failure/sec$ (f/s), i.e., $\lambda = R_{SEU} \times N_{CB}$. where r_{SEU}

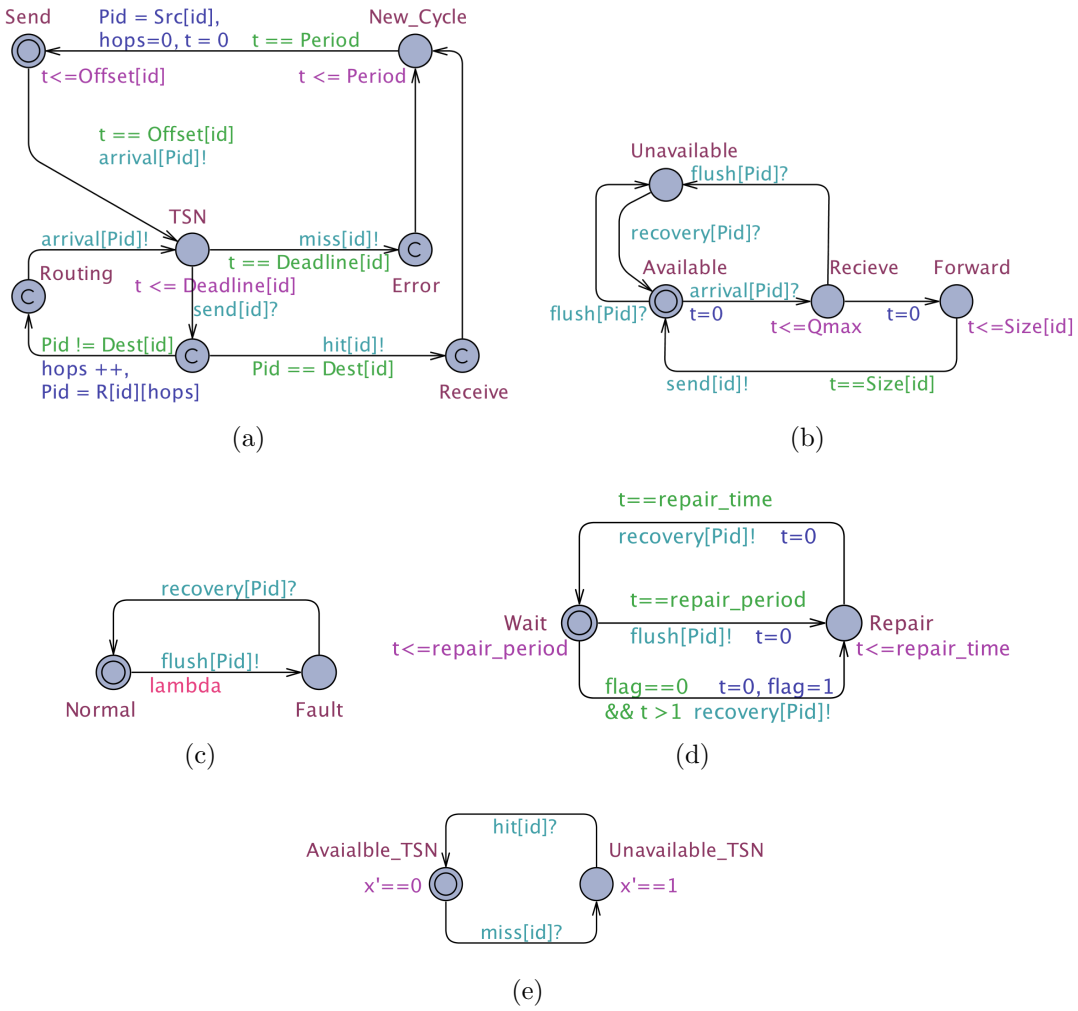


Figure 6.2: Proposed TA model of (a) a critical stream, (b) each output port, (c) the fault injection module, (d) the repair controller, (e) the availability observer.

is the SEU rate which depends on the SRAM technology and the radiation levels. N_{CB} is the number of critical bits in the configuration memory. N_{CB} is typically determined by a fault-injection campaign, which is out the scope of our research project. Thus, our model takes λ as an input parameter, and the automaton returns to state `Normal` according to the synchronization channel `recovery[Pid]?` which indicates that the repair process is accomplished. This TA initiates the repair process, as shown in Fig. 6.2(d). At the beginning, the automaton transit at random time $\leq$

`repair_period` to state `Repair`. These transition model the lake of synchronization between the scrubbing of different switches. The flag is updated to 1, which blocks the lower transition until the end of the simulation. Then, for every `repair_period`, the automaton transits to state `Repair` and triggers channel `flush[Pid]!` to get the switch `Pid` out of service during the repair time. Afterward, the automaton stays for time interval `repair_time` before triggering channel `recovery[Pid]!` and moving to state `Wait`.

6.3 Evaluation

6.3.1 TSN Reliability Analysis Using Model Checking

In this section, we validate the proposed framework. Hence, we consider one synthetic test case and a realistic case study from the space area based on the Ethernet-based network in the ORION CEV (McCabe & Baggerman, 2009).

The synthetic test cases comprise 25 and 50 TT messages, respectively, routed through a network topology that comprises 12 ports deployed in 5 bridges, as shown in Fig. 6.3. The network model is implemented using the UPPAAL model checker (Larsen et al., 1997). Each message's source and destination are randomly assigned from the list of ES. The transmission time of the messages is between 2 and 10 μs. All messages have the same deadlines and a period of 50 μs and 200 μs, respectively. Three schedules for each test case are synthesized using the greedy heuristic algorithm introduced in (Atallah et al., 2018).

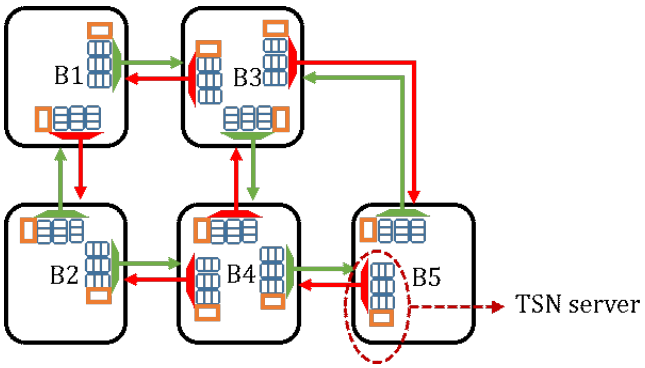


Figure 6.3: Example topology of five-bridges TSN network.

TSN Vulnerability to SEUs For Single-Path Routing

The vulnerability analysis is conducted for single-path routing, i.e., each message is sent through one path selected according to the shortest path criteria. In this analysis, different SEUs are injected to induce accidental Gate Blocking (GB) in CGCV. Results demonstrate that such SEU causes a drop of some messages. This observation can be explained as the following. Since CGCE opens the gate for a time interval equal to the transmission time of the respective TT message, blocking the gate for one CGCE makes the total opening timeless than the required time for transmitting all the messages. Hence, the port is rendered incapable to re-transmit all arrive messages. This situation eventually overflows the buffer and causes some messages to be dropped. The second effect of this type of SEUs is message lagging. The blocked message has to wait in the buffer until the next open time slot to be transmitted. Every message that arrives after GB will be delayed and take the place of the subsequent message. Hence, all messages which are affected by message lagging may miss their deadlines.

The Sensitive Entry (SE) is the entry at which a flip affects the messages delivery in the network. The percentage of SEs, as well as the percentage of Vulnerable Messages (VMs) for three valid schedules for each test case, are shown in Table 6.1.

Table 6.1: SEU Vulnerability For single-path Routing For Different Schedules.

Test case I (25 messages)					
Index	Number of CGCEs	Number of SE	(%)	Number of VM	(%)
S_1	43	40	93%	25	100%
S_2	41	41	100%	23	92%
S_3	52	45	86%	20	80%
Test case II (50 messages)					
Index	Number of CGCEs	Number of SE	(%)	Number of VM	(%)
S_1	84	76	91%	48	96%
S_2	92	79	86%	45	90%
S_3	77	71	92%	50	100%

The results show a high SEU vulnerability in both test cases. These unreliable single-path routing results are expected because this method yields all physical links as a single point of failure. Thus, unplanned gate closing affects the timing of all messages that pass through it.

TSN Vulnerability to SEUs For Multiple-Path Routing

As mentioned before, the TSN standard adapts a path redundancy approach to attain high transmission reliability. Such redundancy is supposed to guarantee an interruption-free fault-resilience in the presence of link failure. To examine this assumption, we applied double redundancy criteria in which every message is transmitted through two disjoint paths, i.e., a failure in a single link should not render a *communication failure*.

In this analysis, different SEUs were injected, and their impact is investigated. Our results demonstrate that reliability is not fully guaranteed. Surprisingly, we observe that the fault injection in some entries leads to communication failures.

Table 6.2 shows the results of the analysis of three different schedules for each

Table 6.2: SEU Vulnerability For MP Routing for Different Schedules.

Test case I (25 messages)					
Index	number of CGCEs	number of SE	(%)	number of VM	(%)
S_1^*	74	0	0%	0	0%
S_2^*	82	2	2.5%	2	8%
S_3^*	79	0	0%	0	0%
Test case II (50 messages)					
Index	number of CGCEs	number of SE	(%)	number of VM	(%)
S_1^*	155	5	3.2%	6	12%
S_2^*	146	8	5.4%	10	20%
S_3^*	152	0	0%	0	0%

test case considering double-path routing. It can be observed that even with the multipath redundancy, some of the designs are still vulnerable to SEUs for both test cases. This observation highlights the need for verifying the design after adapting such redundancy to select the robust schedule. However, we can notice that in the test case with the lower utilization (25) messages, two schedules achieve complete tolerance against SEUs in TAS configuration. On the other hand, test case II with higher utilization shows higher vulnerability, i.e., two of three schedules have sensitive entries, which causes many critical TT messages to be vulnerable to SEUs.

6.3.2 Availability Analysis of FPGA-Based TSN

The TA model is implemented with the UPPAAL toolbox (David, Larsen, Legay, Mikučionis, & Poulsen, 2015). The availability analysis results for single path transmission are depicted in Fig. 6.4. These results show the impact of network utilization and the repair period on network availability. We can notice an optimal repair period for each radiation level (λ). In particular, Fig. 6.4 shows an optimal repair period at 1 and 10 seconds for $\lambda = 10^{-4}$, and $\lambda = 10^{-5}$, respectively. On the other hand,

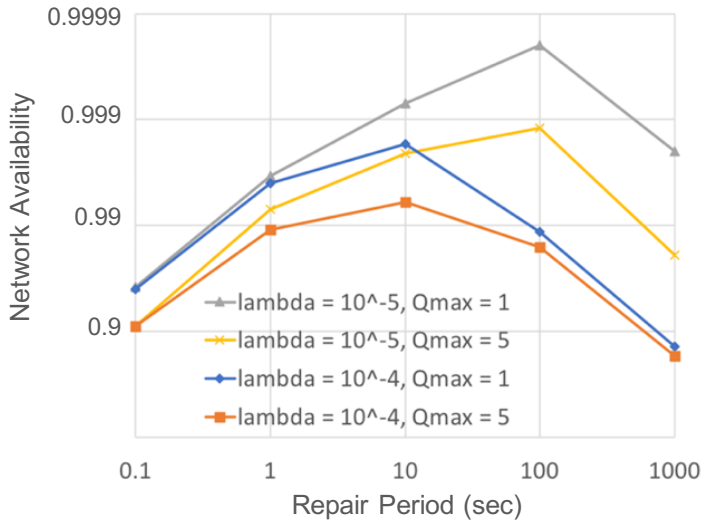


Figure 6.4: Network availability for different failure rates λ and repair periods for single path transmission.

the results show that increasing network utilization declines network availability. For example, The network availability drops from three nines to two nines when Qmax raised from 1 ms to 5 ms at `repair_period` = 100 seconds, $\lambda = 10^{-5}$ f/s. This variation is due to the long time spent through the network. The higher utilization increases the probability of SEU strikes during message transmission.

Safety-critical systems, such as aerospace applications, typically require higher availability levels, e.g., 0.9999 or 0.99999. Such availability levels can still be attained using network-level fault mitigation techniques, such as double path redundancy. Table 6.3 shows the gain of adapting double path redundancy on network availability. The entries of Table 6.3 are obtained according to (57).

$$\mathcal{A}_{doublePath} = 1 - (1 - \mathcal{A}_{singlePath})^2 \tag{57}$$

Table 6.3: Network Availability Under Double Path Redundancy.

Repair period (ms)	$\lambda = 10^{-4}$		$\lambda = 10^{-5}$	
	$Q_{max} = 1$	$Q_{max} = 5$	$Q_{max} = 1$	$Q_{max} = 5$
1000000	0.98	0.97	**0.999996**	0.9996
100000	0.9996	0.9994	**0.9999997**	**0.999998**
10000	**0.999997**	0.99996	**0.999995**	**0.999995**
1000	0.999981	0.9998	**0.999991**	0.99993
100	0.9984	0.992	0.998	0.992

Table 6.3 depicts the estimated network availability, which passes the five nines threshold in bold, which implies 5 minutes of downtime per year. Under the harsh radiation conditions ($\lambda = 10^{-4} f/s$) the network is able to attain an availability > 0.99999 only at `repair_period` = 1 second and at Qmax = 1 ms, while the five nines availability is infeasible for Qmax = 5 ms. This result suggests increasing the network bandwidth to avoid this highly utilized condition. These results demonstrate the tradeoff between the network utilization, the repair period, and the network's availability. Moreover, the results highlight the importance of adapting multipath redundancy in safety-critical applications.

Chapter 7

Conclusion and Future Work

The emerging deployment of CPSs in safety-critical applications imposes challenges to the communication technologies. Stringent reliability and timing constraints must be satisfied under tight cost, weight, and power budgets. Facing such challenges is an opportunity for real-time communication engineers to develop novel methods to design networks with high performance, utilization, and flexibility. One promising solution for such applications is the IEEE 802.1 TSN standard. This thesis aims to pave the way for employing TSN on large-scale systems. This goal is achieved by introducing a new framework for the design and verification of TSN for safety-critical CPS. Several methods are proposed to address the design challenges and meet the requirements of a broad spectrum of CPS. Our framework provides automated design space exploration to support system engineers in making design decisions.

To meet a broad spectrum of reliability constraints in different CPSs, we introduce two methods that employ spatial and temporal redundancy approaches. First, we propose a scalable heuristic-based method for fault-tolerant topology synthesis. This method ensures many disjoint paths between sources and destinations while considering the timing requirements of TT traffic. The redundancy level can be

adjusted based on the criticality of the traffic. The second method is based on ILP formulations for reliability-aware AVB traffic routing. The formulations support both spatial or temporal redundancy to meet the reliability constraints for different CPSs. In particular, the spacial redundancy meets high critical application requirements by providing seamless fault tolerance against physical failures, e.g., downlinks, and soft error, e.g., corrupted packets. The temporal redundancy suits relaxed applications in which the system tolerates short faults waiting for some recovery mechanisms.

To attain real-time communication, we introduce a novel method for scalable scheduling of TT traffic. The proposed method can handle large networks to meet the requirements of the upcoming realistic CPSs. Our method generates no-wait schedules in which the critical TT messages are immediately forwarded over switches without suffering any queuing delay. An iterated ILP-based scheduling is adopted for scalability while the degree of conflict between iteration is minimized using a graph-based technique. To optimize the TSN design for Fog application, we proposed two reinforcement learning-based techniques which enable dynamic priority assignment and speed up the design space exploration.

To verify the TSN reliability under a harsh environment, we introduce a formal model to investigate the impact of the SEU-induced faults. The proposed methods employ a statistical model checking approach to analyze the network behavior under a harsh radiation environment. Two models are introduced to address both TT and RC traffic classes. Our work investigates several design aspects such as the impact of faulty TAS, and the optimum repair frequencies for SRAM-based FPGA TSN switches. The analysis reveals the worst-case failure pattern, which allows further worst-case stability and performance analysis. Such an analysis allows dismissing vulnerable designs at early stages. Extensive experimental results are introduced to evaluate each element of the proposed framework.

Several future directions can be considered based on this thesis which addresses the timing and reliability challenges of TSN at a high level of abstraction. An interesting direction is to investigate and compare different optimization techniques for TSN design space exploration. A better understanding of each technique's pros and cons will diversify the design options and address a broader spectrum of objectives. Nature-inspired optimization techniques, such as ant colony, bees algorithm, and swarm intelligence, can be studied to boost TSN design in terms of scalability and solution quality. Another research direction is investigating the security aspects in TSN design. The security concerns regarding CPSs are increasing due to their role in countries' national security via safety-critical systems like smart grids and autonomous vehicles, which are already deploying TSN. These concerns are addressed by new security measures that generate extra overheads and limit the flexibility of TSN. Therefore, it is critical to investigate the impact of such security-related overheads on the TSN performance.

Publications

Refereed Journals

- **Jr1** Atallah, A., Bany Hamad, G., & Ait Mohamed, O. (2019). Routing and scheduling of time-triggered traffic in time-sensitive networks. IEEE Transactions on Industrial Informatics, 16(7), 4525-4534. (**ISI Q1, Impact Factor 9.112**).

- **Jr2** Atallah, A., Bany Hamad, G., & Ait Mohamed, O. Using Reinforcement Learning for Dynamic Priority Assignment in Time Sensitive Networks, **submitted** to IEEE Transactions on Industrial Informatics on **Dec 10, 2020** (Manuscript ID: TII-20-5566).

Refereed Conferences

- **Cf1** Atallah, A., Bany Hamad, G., & Ait Mohamed, O. (2019, July). Multi-path routing of mixed-critical traffic in time sensitive networks. In International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (pp. 504-515). Springer, Cham. **(Best Paper)**

- **Cf2** Atallah, A., Bany Hamad, G., Ait Mohamed, O., & Boukadoum, M. (2019, June). High-Level Availability Analysis of FPGA-Based Time-Sensitive Networks. In International New Circuits and Systems Conference (NEWCAS) (pp. 1-4). IEEE.

- **Cf3** Atallah, A., Bany Hamad, G., & Ait Mohamed, O. (2019, March). Reliability analysis of TSN networks under SEU induced soft error using model checking. In IEEE Latin American Test Symposium (LATS) (pp. 1-6). IEEE.

- **Cf4** Atallah, A., Bany Hamad, G., & Ait Mohamed, O. (2018, July). Fault-resilient topology planning and traffic configuration for IEEE 802.1 Qbv TSN networks. In International Symposium on On-Line Testing And Robust System Design (IOLTS) (pp. 151-156). IEEE.

- **Cf5** Atallah, A., Bany Hamad, G., & Ait Mohamed, O. (2018, June). Reliability-aware routing of avb streams in tsn networks. In International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (pp. 697-708). Springer, Cham.

- **Cf6** Atallah, A., Bany Hamad, G., & Ait Mohamed, O. (2017, December). Automotive safety verification under temporal failure of adaptive cruise control system using statistical model checking. In International Conference on Embedded & Distributed Systems (EDiS) (pp. 1-6). IEEE.

References

Annighoefer, B., Reif, C., & Thieleck, F. (2014). Network topology optimization for distributed integrated modular avionics. In *Digital avionics systems conference (dasc)* (p. 1-12).

Atallah, A., Bany Hamad, G., & Ait Mohamed, O. (2018). Fault-resilient topology planning and traffic configuration for ieee 802.1qbv tsn networks. In *International symposium on on-line testing and robust system design (iolts)* (p. 1-6).

Bonomi, F., Milito, R., Natarajan, P., & Zhu, J. (2014). Fog computing: A platform for internet of things and analytics. In *Big data and internet of things: A roadmap for smart environments* (pp. 169–186). Springer.

Caffrey, M., Echave, M., Fite, C., Nelson, T., Salazar, A., & Storms, S. (2002). A space-based reconfigurable radio. In *Conference on engineering of reconfigurable systems and algorithms* (pp. 49–53).

Chakraborty, S., Lukasiewycz, M., Buckl, C., Fahmy, S., Chang, N., Park, S., ... Adlkofer, H. (2012). Embedded systems and software challenges in electric vehicles. In *Design, automation and test in europe (date)* (pp. 424–429).

Consortium, O., et al. (2017). Openfog reference architecture for fog computing. *Architecture Working Group*.

Cour, T., Yu, S., & Shi, J. (Copyright 2004). *Normalized cut segmentation code.*

(https://github.com/areslp/matlab/tree/master/Ncut_9, accessed 19-06-03)

Craciunas, S. S., Oliver, R. S., & AG, T. C. (2017). An overview of scheduling mechanisms for time-sensitive networks. *Proceedings of the Real-time summer school LÉcole dÉté Temps Réel (ETR)*.

Craciunas, S. S., Oliver, R. S., Chmelík, M., & Steiner, W. (2016). Scheduling real-time communication in IEEE 802.1Qbv time sensitive networks. In *International conference on real-time networks and systems (rtns)* (pp. 183–192).

David, A., Larsen, K. G., Legay, A., Mikučionis, M., & Poulsen, D. B. (2015). Uppaal smc tutorial. *International Journal on Software Tools for Technology Transfer*, *17*(4), 397–415.

Da Xu, L., He, W., & Li, S. (2014). Internet of things in industries: A survey. *IEEE Transactions on industrial informatics*, *10*(4), 2233–2243.

Decotignie, J.-D. (2005). Ethernet-based real-time and industrial communications. *Proceedings of the IEEE*, *93*(6), 1102–1117.

Desai, N., & Punnekkat, S. (2020). Enhancing fault detection in time sensitive networks using machine learning. In *2020 international conference on communication systems & networks (comsnets)* (pp. 714–719).

Dürr, F., & Nayak, N. G. (2016). No-wait packet scheduling for ieee time-sensitive networks (tsn). In *Proceedings of the 24th international conference on real-time networks and systems* (pp. 203–212).

Etschberger, K. (2001). Controller area network: basics, protocols, chips and applications. *Ixxat press*.

Falk, J., Dürr, F., & Rothermel, K. (2018). Exploring practical limitations of joint routing and scheduling for tsn with ilp. In *Conference on embedded and real-time computing systems and applications* (pp. 136–146).

Finn, N. (2018). Introduction to time-sensitive networking. *IEEE Communications Standards Magazine*, *2*(2), 22–28.

García-Valls, M., Cucinotta, T., & Lu, C. (2014). Challenges in real-time virtualization and predictable cloud computing. *Journal of Systems Architecture*, *60*(9), 726–740.

Gavrilut, V., Zarrin, B., Pop, P., & Samii, S. (2017). Fault-tolerant topology and routing synthesis for ieee time-sensitive networking. In *Proceedings of international conference on real-time networks and systems* (pp. 267–276).

Goyal, P., & Vin, H. M. (1997). Generalized guaranteed rate scheduling algorithms: a framework. *IEEE/ACM transactions on networking*, *5*(4), 561–571.

Gutiérrez, M., Ademaj, A., Steiner, W., Dobrin, R., & Punnekkat, S. (2017). Self-configuration of IEEE 802.1 tsn networks. In *22nd international conference on emerging technologies and factory automation (etfa)* (pp. 1–8).

Gutiérrez, M., Steiner, W., Dobrin, R., & Punnekkat, S. (2015). Learning the parameters of periodic traffic based on network measurements. In *2015 ieee international workshop on measurements & networking (m&n)* (pp. 1–6).

Hamza, T., Scharbarg, J.-L., & Fraboul, C. (2014). Priority assignment on an avionics switched ethernet network (qos afdx). In *Workshop on factory communication systems* (pp. 1–8).

Haur, N. K., & Chin, T. S. (2019). Challenges and future direction of time-sensitive software-defined networking (tssdn) in automation industry. In *International conference on security, privacy and anonymity in computation, communication and storage* (pp. 309–324).

Heise, P., Geyer, F., & Obermaisser, R. (2016). Tsimnet: An industrial time sensitive networking simulation framework based on omnet++. In *New technologies, mobility and security (ntms), 2016 8th ifip international conference on* (pp.

1–5).

IEEE Standard for Local and metropolitan area networks–Bridges and Bridged Networks (Standard). (2014). USA: IEEE.

Imtiaz, J., Jasperneite, J., & Weber, K. (2012). Approaches to reduce the latency for high priority traffic in ieee 802.1 avb networks. In *2012 9th ieee international workshop on factory communication systems* (pp. 161–164).

ISO26262, Road vehicles – Functional Safety – Part 1-9, 1st edition. (Standard). (2011). International Organization for Standardization.

Jeon, S., Lee, J., & Park, S. (2015). Dual-path method for enhancing the performance of ieee 802.1 avb with time-triggered scheme. In *Communications (apcc), 2015 21st asia-pacific conference on* (pp. 519–523).

Kanabar, M. G., & Sidhu, T. S. (2009). Reliability and availability analysis of iec 61850 based substation communication architectures. In *Power & energy society general meeting, 2009. pes'09. ieee* (pp. 1–8).

Kauer, M., Soudbakhsh, D., Goswami, D., Chakraborty, S., & Annaswamy, A. M. (2014). Fault-tolerant control synthesis and verification of distributed embedded systems. In *Proceedings of the conference on design, automation & test in europe* (p. 56).

Kehrer, S., Kleineberg, O., & Heffernan, D. (2014). A comparison of fault-tolerance concepts for ieee 802.1 time sensitive networks (tsn). In *Emerging technology and factory automation (etfa)* (pp. 1–8).

Kopetz, H. (2011). *Real-time systems: design principles for distributed embedded applications*. Springer Science & Business Media.

Larsen, K., Pettersson, P., & Yi, W. (1997). Uppaal in a nutshell. *International journal on software tools for technology transfer*, *1*(1-2), 134–152.

Laursen, S. M., Pop, P., & Steiner, W. (2016). Routing optimization of avb streams

in tsn networks. *ACM SIGBED Review*, *13*(4), 43–48.

Lee, E. (2008). Cyber physical systems: Design challenges. In *Ieee international symposium on object oriented real-time distributed computing isorc* (pp. 363–369).

Leppinen, H., Kestilä, A., Tikka, T., & Praks, J. (2016). The aalto-1 nanosatellite navigation subsystem: Development results and planned operations. In *2016 european navigation conference (enc)* (pp. 1–8).

Li, Z., Wan, H., Deng, Y., Zhao, X., Gao, Y., Song, X., & Gu, M. (2018). Time-triggered switch-memory-switch architecture for time-sensitive networking switches. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*.

Lukasiewycz, M., Schneider, R., Goswami, D., & Chakraborty, S. (2012). Modular scheduling of distributed heterogeneous time-triggered automotive systems. In *Design automation conference (asp-dac), asia and south pacific* (pp. 665–670).

Mahfuzi, R., Aminifar, A., Samii, S., Rezine, A., Eles, P., & Peng, Z. (2018a). Stability-aware integrated routing and scheduling for control applications in ethernet networks. In *Design, automation and test in europe (date).*

Mahfuzi, R., Aminifar, A., Samii, S., Rezine, A., Eles, P., & Peng, Z. (2018b). Stability-aware integrated routing and scheduling for control applications in ethernet networks. In *Date.*

Mai, T. L., Navet, N., & Migge, J. (2019). *A hybrid machine learning and schedulability method for the verification of tsn networks* (Tech. Rep.). University of Luxembourg.

Makowitz, R., & Temple, C. (2006). Flexray-a communication network for automotive control systems. In *2006 ieee international workshop on factory communication systems* (pp. 207–212).

Margaria, T., & Steffen, B. (2010). *Leveraging applications of formal methods, verification, and validation: International symposium on leveraging applications, proceedings, part i* (Vol. 6415). Springer.

Maxim, D., & Song, Y.-Q. (2017). Delay analysis of avb traffic in time-sensitive networks (tsn). In *Proceedings of the 25th international conference on real-time networks and systems* (pp. 18–27).

McCabe, M., & Baggerman, C. (2009). Avionics architecture interface considerations between constellation vehicles. In *Ieee/aiaa digital avionics systems conference,* (pp. 1–E).

McCabe, M., Baggerman, C., & Verma, D. (2009). Avionics architecture interface considerations between constellation vehicles. In *2009 ieee/aiaa 28th digital avionics systems conference* (pp. 1–E).

Messenger, J. L. (2018). Time-sensitive networking: An introduction. *IEEE Communications Standards Magazine*, *2*(2), 29–33.

Morgan, P. S. (2016). Enhancing the cassini mission through fp applications after launch. In *2016 ieee aerospace conference* (pp. 1–20).

Navet, N., Mai, T. L., & Migge, J. (2019). *Using machine learning to speed up the design space exploration of ethernet tsn networks* (Tech. Rep.). University of Luxembourg.

Nayak, N. G., Duerr, F., & Rothermel, K. (2018). Routing algorithms for ieee802. 1qbv networks. *ACM SIGBED Review*, *15*(3), 13–18.

Nayak, N. G., Dürr, F., & Rothermel, K. (2018). Incremental flow scheduling and routing in time-sensitive software-defined networks. *IEEE Transactions on Industrial Informatics*, *14*(5), 2066–2075.

Normand, E. (1996). Single event upset at ground level. *IEEE transactions on Nuclear Science*, *43*(6), 2742–2750.

Norton, C. D., Werne, T. A., Pingree, P. J., & Geier, S. (2009). An evaluation of the Xilinx Virtex-4 FPGA for on-board processing in an advanced imaging system. In *Aerospace conference* (pp. 1–9).

Ojewale, M. A., & Yomsi, P. M. (2020). Routing heuristics for load-balanced transmission in tsn-based networks. *ACM Sigbed Review*, *16*(4), 20–25.

Pahlevan, M., Schmeck, J., & Obermaisser, R. (2019). Evaluation of tsn dynamic configuration model for safety-critical applications. In *2019 ieee intl conf on parallel & distributed processing with applications, big data & cloud computing, sustainable computing & communications, social computing & networking (ispa/bdcloud/socialcom/sustaincom)* (pp. 566–571).

Papadopoulos, C. (2004). An automata-based approach to cscw verification. *International Journal of Cooperative Information Systems*, *13*(02), 183–209.

Paulitsch, M., Schmidt, E., Gstöttenbauer, B., Scherrer, C., & Kantz, H. (2011). Time-triggered communication (industrial applications). *Time-Triggered Communication*, 121–152.

Poovendran, R. (2010). Cyber–physical systems: Close encounters between two parallel worlds [point of view]. *Proceedings of the IEEE*, *98*(8), 1363–1366.

Pop, P., Raagaard, M. L., Craciunas, S. S., & Steiner, W. (2016). Design optimisation of cyber-physical distributed systems using ieee time-sensitive networks. *IET Cyber-Physical Systems: Theory & Applications*, *1*(1), 86–94.

Pop, P., Raagaard, M. L., Gutierrez, M., & Steiner, W. (2018). Enabling fog computing for industrial automation through time-sensitive networking (tsn). *IEEE Communications Standards Magazine*, *2*(2), 55–61.

Pozo, F., Rodriguez-Navas, G., Hansson, H. A., & Steiner, W. (2017). *Schedule synthesis for next generation time-triggered networks.* Malardalen Real-Time Research Centre.

Pozo, F., Steiner, W., Rodriguez-Navas, G., & Hansson, H. (2015). A decomposition approach for smt-based schedule synthesis for time-triggered networks. In *Emerging technologies & factory automation (etfa), 2015 ieee 20th conference on* (pp. 1–8).

Raagaard, M. L., Pop, P., Gutiérrez, M., & Steiner, W. (2017). Runtime reconfiguration of time-sensitive networking (tsn) schedules for fog computing. In *2017 ieee fog world congress (fwc)* (pp. 1–6).

Schaul, T., Quan, J., Antonoglou, I., & Silver, D. (2015). Prioritized experience replay. *arXiv preprint arXiv:1511.05952*.

Schweissguth, E., Danielis, P., Timmermann, D., Parzyjegla, H., & Mühl, G. (2017). ILP-based joint routing and scheduling for time-triggered networks. In *Proceedings of international conference on real-time networks and systems* (pp. 8–17).

Shi, J., & Malik, J. (2000). Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, *22*(8), 888–905.

Siegle, F., Vladimirova, T., Ilstad, J., & Emam, O. (2016). Availability analysis for satellite data processing systems based on SRAM FPGAs. *IEEE Transactions on Aerospace and Electronic Systems*, *52*(3), 977–989.

Smirnov, F., Glaß, M., Reimann, F., & Teich, J. (2016). Formal reliability analysis of switched ethernet automotive networks under transient transmission errors. In *Acm/edac/ieee design automation conference (dac)* (pp. 1–6).

Smirnov, F., Glaß, M., Reimann, F., & Teich, J. (2017a). Formal timing analysis of non-scheduled traffic in automotive scheduled tsn networks. In *2017 design, automation & test in europe conference & exhibition (date)* (pp. 1643–1646).

Smirnov, F., Glaß, M., Reimann, F., & Teich, J. (2017b). Optimizing message routing and scheduling in automotive mixed-criticality time-triggered networks. In *Design automation conference (dac)* (pp. 1–6).

Smirnov, F., Reimann, F., Teich, J., Han, Z., & Glaß, M. (2018). Automatic optimization of redundant message routings in automotive networks. In *Proceedings of the 21st international workshop on software and compilers for embedded systems* (pp. 90–99).

Specht, J., & Samii, S. (2016). Urgency-based scheduler for time-sensitive switched ethernet networks. In *2016 28th euromicro conference on real-time systems (ecrts)* (pp. 75–85).

Specht, J., & Samii, S. (2017). Synthesis of queue and priority assignment for asynchronous traffic shaping in switched ethernet. In *2017 ieee real-time systems symposium (rtss)* (pp. 178–187).

Steiner, W., Craciunas, S. S., & Oliver, R. S. (2018). Traffic planning for time-sensitive communication. *IEEE Communications Standards Magazine*, *2*(2), 42–47.

Swift, G., Carmichael, C., Allen, G., Madias, G., Miller, E., Monreal, R., et al. (2011). Compendium of xrtc radiation results on all single-event effects observed in the virtex-5qv. *Proceedings of NASA military and aerospace programmable logic devices (MAPLD)*, 1–33.

Tămaş-Selicean, D., Pop, P., & Steiner, W. (2015). Design optimization of ttethernet-based distributed real-time systems. *Real-Time Systems*, *51*(1), 1–35.

Tanasa, B., Bordoloi, U. D., Eles, P., & Peng, Z. (2010). Scheduling for fault-tolerant communication on the static segment of flexray. In *Real-time systems symposium (rtss)* (pp. 385–394).

Tindell, K. W., Burns, A., & Wellings, A. J. (1992). Allocating hard real-time tasks: an np-hard problem made easy. *Real-Time Systems*, *4*(2), 145–165.

TTTech. (2014). *Systems integration with deterministic ethernet: Nasa's orion multipurpose crew vehicle* (Standard). The TTTech Company.

Van Hasselt, H., Guez, A., & Silver, D. (2016). Deep reinforcement learning with double q-learning. In *Thirtieth aaai conference on artificial intelligence* (pp. 1–7).

Volodymyr, M., Koray, K., David, S., Andrei, A. R., & Joel, V. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533.

Wang, B., & Hou, J. C. (2000). Multicast routing and its qos extension: problems, algorithms, and protocols. *IEEE network*, *14*(1), 22–36.

Wang, C., Li, J., & Hu, F. (2011). Fault tree synthesis for an avionic networksimulation-network-daud. In *Transportation, mechanical, and electrical engineering (tmee), 2011 international conference on* (pp. 155–159).

Wang, Y., Wang, K., Huang, H., Miyazaki, T., & Guo, S. (2018). Traffic and computation co-offloading with reinforcement learning in fog computing for industrial applications. *IEEE Transactions on Industrial Informatics*, *15*(2), 976–986.

Wu, Z., & Leahy, R. (1993). An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, *15*(11), 1101–1113.

Yen, J. Y. (1970). An algorithm for finding shortest routes from all source nodes to a given destination in general networks. *Quarterly of Applied Mathematics*, *27*(4), 526–530.

Yoo, I., Jo, J., Ju, Y., & Park, I.-C. (2017). Unidirectional ring ethernet and media access controller with automatic relaying for low-complexity in-vehicle control network. *Journal of Semiconductor Technology and Science*, *17*(5), 697–708.

Zhang, H., & Ferrari, D. (1993). Rate-controlled static-priority queueing. In *Ieee infocom'93 the conference on computer communications, proceedings* (pp. 227–236).