On Preemption in Time-Sensitive Networking (TSN)

Mubarak Adetunji Ojewale

Supervisor: Dr. Patrick Meumeu Yomsi

Co-Supervisor: Prof. Dr. Luís Miguel Pinho de Almeida

Programa Doutoral em Engenharia Electrotécnica e de Computadores

February, 2023

Faculdade de Engenharia da Universidade do Porto

On Preemption in Time-Sensitive Networking (TSN)

Mubarak Adetunji Ojewale

Dissertation submitted to Faculdade de Engenharia da Universidade do Porto
to obtain the degree of

Doctor Philosophiae in Electronic & Computer Engineering

President: Prof. Dr. Manuel Alberto Pereira Ricardo
Referee: Prof. Dr. Saad Mubeen
Referee: Dr. Ramon Serna Oliver
Referee: Prof. Dr. Mário Jorge Rodrigues de Sousa
Referee: Prof. Dr. Paulo Bacelar Reis Pedreiras
Referee: Prof. Dr. Pedro Alexandre Guimarães Lobo Ferreira Souto
Supervisor: Dr. Patrick Meumeu Yomsi

February, 2023

Abstract

A key requirement for responsiveness, i.e., timely and correct response to events, in real-time distributed applications is the ability of data to move in a reliable and predictable manner across the underlying communication network. Ethernet is the emerging communication technology for modern real-time wired networks. Its ability to meet the increasingly stringent speed and distance requirements, along with its high bandwidth capacity, makes it a promising addition to legacy communication infrastructures. The older Ethernet standards were originally geared towards non-real-time applications and lacked features to support real-time communication. To close this gap, the Institute of Electrical and Electronics Engineers (IEEE) has proposed several changes to standards over the past three decades. The collection of recent standards introduced for this purpose is called Time Sensitive Networking (TSN). Among all these, the IEEE 802.1Qbu, which specifies a 1-level frame preemption mechanism, occupies a prominent place. Specifically, the TSN frame preemption mechanism is specified by the so-called 1-*level preemption scheme* as follows. Frames are divided into two classes: (*i*) the *express frames*, which are considered urgent and therefore eligible for expedited transmission, and (*ii*) the *preemptable frames*, which are considered less urgent. In particular, express frames can preempt preemptable frames and two frames of the same class cannot preempt each other.

While frame preemption significantly improves the suitability of Ethernet for real-time communication with strict and heterogeneous requirements, the current mode of operation, as specified by TSN, has some serious limitations that affect network performance. Most importantly, this scheme is prone to performance degradation when the number of express frames is high. In addition, preemptable frames with firm timing requirements can suffer from long blocking periods due to priority inversions, since frames in the same preemption class cannot preempt each other. These limitations mean that the 1-level preemption scheme does not provide a way to efficiently support the coexistence of flows with diverse timing requirements on the same network. In this thesis, we show the limitations of 1-level preemption in real-time applications. We then postulate that these shortcomings can be effectively mitigated with a **multi-level preemption scheme**, and we show the feasibility of this scheme as well as its requirements, timing analysis, and configuration. We make a fourfold contribution: (1) We propose a new framework in which the non-preemptive transmission constraints between non-express frames are relaxed. Then, we describe the operational dynamics of our approach and the actual implementation recommendations for its feasibility; (2) we perform a comprehensive and rigorous worst-case traversal time (WCTT) analysis for each flow in the network

and compare the results with the 1-level preemption and the non-preemptive schemes; (3) we provide an offline priority assignment scheme for the flow set. Then, we provide an offline framework for determining the appropriate number of preemption levels on the one hand and assigning flows to preemption classes on the other; finally (4) we evaluate the performance gain of a multi-level priority scheme over a 1-level preemption scheme from both qualitative and quantitative perspectives. Several studies have pointed out that the major limitation of frame preemption is that it only allows one level of preemption. By addressing this limitation, this work positions Ethernet TSN with Frame Preemption as a simpler and more cost-effective alternative communication solution for Distributed Real-time Embedded Systems.

Keywords: Time Sensitive Networking, Real-time Communication, Ethernet, Frame Preemption.

Resumo

Um requisito fundamental para capacidade de resposta, ou seja, resposta oportuna e correta a eventos, em aplicativos distribuídos em tempo real, é a capacidade dos dados de se moverem de maneira confiável e previsível pela rede de comunicação subjacente. Ethernet é a tecnologia de comunicação emergente para redes cabeadas em tempo real modernas. Sua capacidade de atender aos requisitos cada vez mais rigorosos de velocidade e distância, juntamente com sua alta capacidade de largura de banda, o torna uma adição promissora às infra-estruturas de comunicação legadas. Os padrões Ethernet mais antigos foram originalmente voltados para aplicativos não em tempo real e careciam de funções dependentes do tempo. Para fechar essa lacuna, o Instituto de Engenheiros Elétricos e Eletrônicos (IEEE) propôs várias mudanças nos padrões nas últimas três décadas. A coleção de padrões recentes introduzidos para essa finalidade é chamada de Time Sensitive Networking (TSN). Entre todos eles, o IEEE 802.1Qbu, que especifica um mecanismo de preempção de quadro de nível 1, ocupa um lugar de destaque. Especificamente, o mecanismo de preempção do quadro TSN é especificado pelo chamado esquema de preempção de 1 nível como segue. Os frames são divididos em duas classes: (i) *os frames expressos*, que são considerados urgentes e, portanto, elegíveis para transmissão acelerada, e (ii) *os frames preemptivos*, que são considerados menos urgentes. Em particular, os quadros expressos podem ter preempção de quadros preemptivos e dois quadros da mesma classe não podem ter preempção um do outro.

Embora a preempção de quadro melhore muito a adequação da Ethernet para comunicação em tempo real com requisitos rígidos e heterogêneos, o modo de operação atual, conforme especificado pelo TSN, possui algumas limitações sérias que afetam o desempenho da rede. Mais importante ainda, esse esquema é propenso à degradação do desempenho quando o número de quadros expressos é alto. Além disso, quadros preemptivos com requisitos de temporização firmes podem sofrer longos períodos de bloqueio devido a inversões de prioridade, uma vez que quadros na mesma classe de preempção não podem preempção uns dos outros. Essas limitações significam que o esquema de preempção de 1 nível não fornece uma maneira eficiente de suportar a coexistência de fluxos com diversos requisitos de tempo na mesma rede. Nesta tese, mostramos as limitações da preempção de 1 nível em aplicações de tempo real.

Em seguida, postulamos que essas deficiências podem ser efetivamente mitigadas com um **esquema de preempção multinível** e mostramos a viabilidade desse esquema, bem como seus requisitos, análise de tempo e configuração. Fazemos uma contribuição quádrupla: (1) Propomos uma nova estrutura na qual as restrições de transmissão não pre-

emptiva entre quadros não expressos são relaxadas. Em seguida, descrevemos a dinâmica operacional de nossa abordagem e as recomendações efetivas de implementação para sua viabilização; (2) realizamos uma análise abrangente e rigorosa do tempo de travessia do pior caso (WCTT) para cada fluxo na rede e comparamos os resultados com a preempção de 1 nível e os esquemas não preemptivos; (3) fornecemos um esquema de atribuição de prioridade off-line para o conjunto de fluxo. Em seguida, fornecemos uma estrutura off-line para determinar o número apropriado de níveis de prioridade, por um lado, e atribuir fluxos a classes de prioridade, por outro; finalmente (4) avaliamos o ganho de desempenho de um esquema de prioridade multinível em relação a um esquema de preempção de nível 1 de ambas as perspectivas qualitativa e quantitativa. Vários estudos apontaram que a principal limitação da preempção de quadros é que ela permite apenas um nível de preempção. Ao abordar esta limitação, este trabalho posiciona a Ethernet TSN com Frame Preemption como uma solução de comunicação alternativa mais simples e econômica para Sistemas Embarcados Distribuídos em Tempo Real.

Palavras-chave: Time Sensitive Networking, Real-time Communication, Ethernet, Frame Preemption.

Acknowledgments

All praise is due to Allah - the Exceedingly Gracious and Exceedingly Merciful. May His peace and blessings be upon His Messenger – Muhammad, his household, his companions and all those who follow the path of guidance. After that, I would like to thank my parents "Baba Oje" and "Iya Oje," as we affectionately call them. The whole thesis is not enough to tell the story of the sacrifices you made so that I could come this far. I would like to thank my siblings Abu Zainab, Umm Zayd, RidwanuLlah, Ashraf, and RahmotaLlah. Thank you for always having my back and for always being there for me.

I would also like to thank all my teachers, mentors, and managers, starting with my Ph.D. advisor, Patrick, who helped me a lot and made sure that I learn everything I need to become an excellent researcher. My special thanks to you, Patrick, for the late nights, the long zoom calls, and for the life-saving purchase at Mar Shopping when I arrived! I must especially mention my co-supervisor, Luis Almeida, up there as one of the nicest human beings I have ever come across, and one of the best in giving honest & objective feedback.

I would like to thank CISTER Director, Eduardo Manuel Medicis Tovar, for creating such a truly international environment. I especially thank you for the prayer room, it made my life easier in many ways than you can imagine. I also thank the members of the Jury who made time out of their busy schedules to evaluate this dissertation.

I would like to thank my colleagues at CISTER for their support and companionship. In no particular order, Konstatinos, Ali, Ramiro, Jose, Dacid, Aftab, Harrison, Miguel, Ishfaq, Yousef, Jatin, Radha, Yilian, Javier, Enio, João, Giann, Jingjing, AbdulHakeem, Saeed, Alam, and Gowar, I say thank you all for the nice memories. I especially thank Cristiana, Inês, Sandra, Marwin for their immense support and kindness.

I would also like to acknowledge the Nigerian community in Porto, who have provided the warmth of home and have made Porto a bit less foreign. The Sadiqs. the Shamsudeens, the Akinsanmis, the Aboderins, the Adurojas, the AbdurRahmans, the Aderojus, the Ogodos, Benjamen, Etubi, and Barikisu, thank you all. I hope I have not left any name. I thank my non-Nigerian friends in Porto, from my good neighbor Zahid, to the Harrisons, Kashif, Inyass, Asif, Motaz and everyone. Thank you all for providing the needed break from the rigors of research. To my friends in Nigeria who bore with my unavailability for long periods, and made our relationship survive this journey, I cherish you so much. The Popoolas, the Adeosuns, Oyetunjis, the Kolawoles, the Sakas, the Adegokes, the Salamis, the Buharis, the Annafis, the Ahmeds, the Sulaimans, the Ismails, the Sanusis, the Busaris, TK Salami, the Oyewos, SM Kester, and everyone, thank you so much. To my wonderful mother-in-law who would check up on us very frequently, I especially acknowledge your kindness and mindfulness.

Finally, I thank my lovely wife and my cute little Sumayyah for their support throughout this journey. You are the ones who truly lived through everything together with me. Thank you for your patience, understanding, and the joy that you bring.

Cheers to the beginning of more research!

This work was partially supported by the project Safe Cities - Inovação para Construir Cidades Seguras, ref. POCI-01-0247-FEDER-041435, co-funded by the European Regional Development Fund (ERDF), through the Operational Programme for Competitiveness and Internationalization (COMPETE 2020); also by National Funds through FCT/MCTES (Portuguese Foundation for Science and Technology), within the CISTER Research Unit (UIDB/04234/2020).

Mubarak Adetunji Ojewale

Publications

1. **M. A. Ojewale**, P. Meumeu Yomsi, and L. Almeida, *A Configuration Framework for Multi-level Preemption Schemes in Time Sensitive Networking*, 30th International Conference on Real-Time Networks and Systems (RTNS 2022). Association for Computing Machinery, New York, NY, USA, 219–229. https://doi.org/10.1145/3534879.3534891.

2. A. Pruski, **M. A. Ojewale**, V. Gavrilut, P. Meumeu Yomsi, M. S. Berger and L. Almeida, *Implementation Cost Comparison of TSN Traffic Control Mechanisms*, 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), 2021, pp. 01-08, doi: 10.1109/ETFA45728.2021.9613463.

3. **M. A. Ojewale**, P. Meumeu Yomsi, B. Nikolić, *Worst-case traversal time analysis of TSN with multi-level preemption*, Journal of Systems Architecture (JSA), Volume 116, 2021, 102079, ISSN 1383-7621, https://doi.org/10.1016/j.sysarc.2021.102079.

4. **M. A. Ojewale**, P. Meumeu Yomsi and B. Nikolić, *Multi-Level Preemption in TSN: Feasibility and Requirements Analysis*, IEEE 23rd International Symposium on Real-Time Distributed Computing (ISORC), 2020, pp. 47-55, doi: 10.1109/ISORC49007.2020.00017.

5. **M. A. Ojewale** and P. Meumeu Yomsi, *Routing heuristics for load-balanced transmission in TSN-based networks*, SIGBED Rev. 16, 4 (December 2019), 20–25. https://doi.org/10.1145/3378408.3378411.

6. **M. A. Ojewale**, P. Meumeu Yomsi, G. Nelissen, *On Multi-level Preemption in Ethernet*, Work-in-Progress (WiP) Session, 30th Euromicro Conference on Real-Time Systems (ECRTS), pp. 16-18, 2018.

Contents

List of Figures

List of Tables

List of Abbreviations

AFXD	Avionics Full Duplex Switched Ethernet
CLB	Audley's Optimal Priority Assignment
AVB	Audio Video Bridge
BE	Best Effort
BLS	Burst Limiting Shaper
bpflows	Best Effort Preemptable Traffic
BRAM	Block Random Access Memory
CAD	Computer-Aided Design
CAN	Controller Area Network
CBS	Credit-Based Shapers
CDT	Control data Traffic
CLB	Configurable Logic Blocks
CoS	Classes of service
CPA	Compositional Performance Analysis
CR-DRR	Congestion Recovering, Dynamic and Replication-aware Routing
DA	Destination Address
DMPO	Deadline-Monotonic Priority Ordering
DRES	Distributed Real-time Embedded Systems
DSG	Digital Signal Processing
ECU	Engine Control Unit
FCS	Frame Check Sequence
FIFO	First-in, First-out
FPGA	Field Programmable Gate Array
eMAC	express Medium Access Control
Frag	Fragment
GM	Grand Master
HDL	Hardware Description Language
IEEE	Institute of Electrical and Electronics Engineers
IFG	Inter Frame Gap
ILP	Integer Linear Programming

IT	Information Technology
LB-DRR	Load-Balanced, Dynamic and Replication-aware Routing algorithm
LIN	Local Interconnect Network
LUT	Look-Up Tables
MAC	Medium Access Control
MCRC	MFrame Cyclic Redundancy Check
ML	Machine Learning
mpflows	Medium Priority Preemptable Traffic
MMS	MAC Merge Sublayer
NC	Network Calculus
NRE	Non-Recurring Engineering
OS	Operating System
OT	Operational Technology
P2P	Point-to-point
PLD	Programmable Logic Device
pMAC	preemptable Medium Acess Control
PS	Peristaltic Shapers
QoS	Quality of service
RC	Rate Constrained
RE	Recurring Engineering
RTES	Real-Time Embedded Systems
RTL	Register Transfer Level
RQ	Research Question
SA	Source Address
SFD	Start Frame Delimiter
SMD-Cx	Start MFrame Delimiter - Continuation Fragment
SMD-E	Start MFrame Delimiter - Express
SMD-Sx	Start MFrame Delimiter - Start Fragment
SMT	Satisfiability Modulo Theories
SRAM	Static Random Access Memory
TA	Trajectory Approach
TAS	Time-Aware Shapers
TSN	Time Sensistive Networking
TSU	Time-Stamping Unit
UBS	Urgency Based Scheduler
VHDL	VHSIC Hardware Definition Language
WCTT	Worst Case Traversal Time

Chapter 1

Introduction

The rise of digital technology has changed people's lives in many ways. Technology components such as sensors, actuators, communication, and computing infrastructures have created an intertwined ecosystem. From traditional systems such as manufacturing, avionics and automotive to emerging systems such as service robots and autonomous systems, digital technology is now at the heart of virtually all human endeavors. Microprocessor technology is driving this new phenomenon. It is becoming ever smaller in miniaturization, yet ever more sophisticated and powerful. Today, microprocessors can be found in everyday devices, from cell phones to home appliances to production systems and aircraft. Such devices/systems that contain microprocessors to provide computation and control functions are often referred to as *embedded systems*. In some application scenarios, such as automotive and industrial automation, an embedded system consists of multiple hardware and software components that must communicate over a network medium. This class of embedded systems is referred to as *distributed embedded systems*. The subset of distributed embedded systems that requires not only functional correctness - i.e., functions must provide logically correct outputs - but also temporal correctness - i.e., these outputs must be available within a specified time interval - is called *Distributed Real-time Embedded Systems (DRES)*. In this work, we focus on the *communication* aspects of DRES. In this kind of systems, different components communicate over a wired or wireless medium, depending on the specific constraints of the target application domain. While wireless communication offers more flexibility and mobility, wired communication is usually preferred because it offers high reliability and is less susceptible to signal loss and interference.

In this dissertation, some key contributions to real-time wired communication in DRES are presented.

The remainder of this chapter is organized as follows. First, we explain the motivation for this research and identify the research gap that we aim to fill in Section 1.2. Then, we present the thesis and the corresponding research question in 1.3. The research methodology is elicited in Section 1.4, which is followed by a summary of the main contributions in Section 1.5. Finally, Section 1.6 explains the significance of the contributions, and Section 1.7 describes the structure of the remainder of the dissertation.

1.1 Research Context

One of the key requirements of DRES is *real-time communication*, i.e., the ability of data to move reliably and predictably through the underlying network. To date, a variety of solutions have been proposed in the literature, each targeting a set of domain-specific requirements. Some examples of established solutions include the *Controller Area Network* (CAN) protocol (Szydlowski, 1992), the *Local Interconnect Network* (LIN) protocol (Specks and Rajnák, 2000), the *FlexRay* protocol (Pop et al., 2006), and TTEthernet (Steiner et al., 2009), which focus mainly on the automotive sector; *Avionics Full Duplex Switched Ethernet* (AFDX) (Brajou and Ricco, 2004), which targets the avionics sector; and protocols such as SERCUS III (Schemm, 2004), EtherCAT (Jansen and Buttner, 2004) and PROFINET (Feld, 2004), which are aimed at the industrial domain. As a matter of fact, most of these solutions are now struggling to keep up with the growing bandwidth and performance requirements of emerging applications in their respective domains (Thiele and Ernst, 2016a). Another development is the convergence of Operations Technology (i.e., the hardware and software systems used to control and monitor industrial processes) and Information Technology (i.e., the systems and tools used to manage, store, and process data), which presents a new challenge and requires new communication technologies to handle different types of traffic (real-time, non-real-time, long and short frames) on the same network infrastructure.

Ethernet Shoch (1981) has emerged as a leading and promising replacement for all previous technologies, capable of meeting increasingly stringent time and distance requirements and providing high bandwidth capacity (Thiele and Ernst, 2016a; Jia et al., 2013). In this protocol, data is transmitted over the network using a digital packet called a ”frame” (see Figure 1.1 for the structure of a basic Ethernet frame). The first field is the ”preamble” – 7 bytes – which consists of alternating 1s and 0s and is used to synchronize the sender and receiver. This field is followed by what is called the “Start Frame Delimiter” – 1 byte –, which indicates the start of the frame; then come the ”Media Access

Control" (MAC) destination and source addresses[1] – 6 bytes each –, which identify the devices sending and receiving the data. These two fields are followed by what is called the "Ethertype" – 2 bytes –, which specifies the type of data being transmitted (e.g., Internet Protocol (IP), Address Resolution Protocol (ARP), etc.). This field is followed by the "Data" field – 46 to 1500 bytes –, which contains the actual data that will be transmitted. Finally, the last field is the "frame check sequence" – 4 bytes – which is used to detect errors during transmission. This is a mathematical calculation that is performed on the frame to ensure that it was transmitted and received correctly.

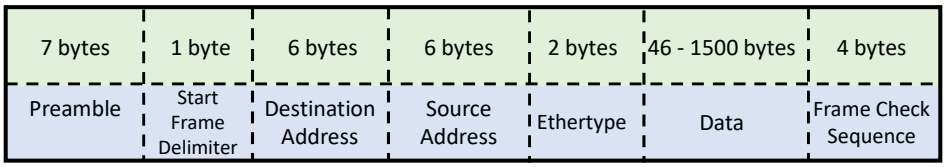


Figure 1.1: Ethernet frame format as defined by the IEEE.

The original Ethernet standards were intended only for non-real-time applications and have several limitations that complicate their use in DRES. In their specification, a frame must not interrupt the transmission of other frames, regardless of its timing constraints. In other words, the frames are transmitted non-preemptively as illustrated in Figure 1.2.

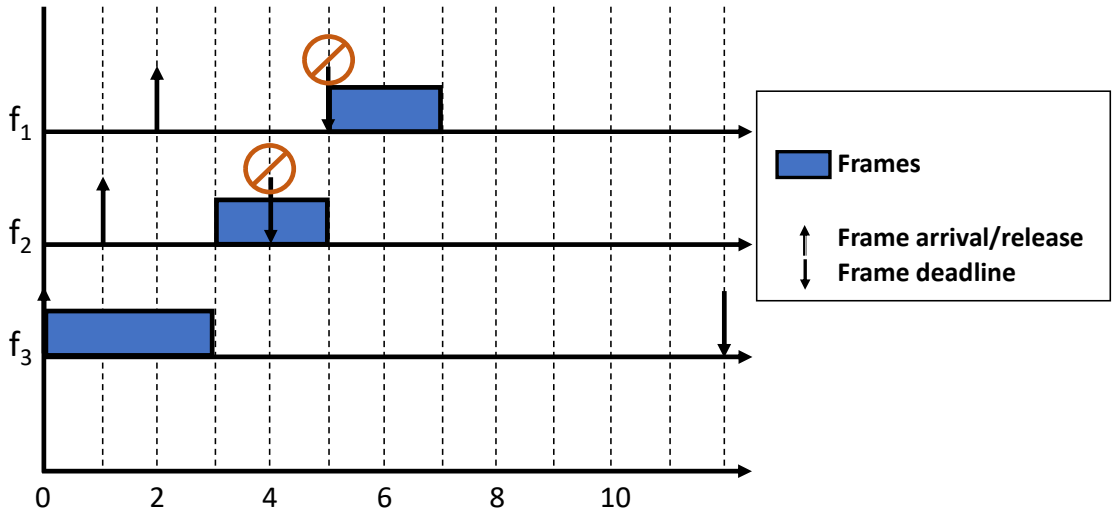


Figure 1.2: Illustration of the Ethernet non-preemptive transmission scheme.

In this figure, three frames (f_1, f_2, and f_3) are considered for transmission in a First-In-First-Out (FIFO) order. The up arrows define the frame arrivals and the down arrows define the time instant by which the transmission of each frame must be completed.

[1]The MAC address is a unique identifier assigned to each device on a network

Frame f_1 arrives first (at time 0) and starts its transmission. During the transmission of f_1, frames f_2 and f_3 arrive, but cannot begin their transmission despite their stringent timing constraint because f_1 is being transmitted. Both frames are given a window of three time units for their transmission. Consequently, they do not meet their timing requirements, i.e., they miss their deadlines. In addition to the potentially large delay incurred by frames due to the non-preemptive transmission scheme, another major limitation of Ethernet when it comes to real-time applications is its unpredictable nature in the sense that it does not guarantee the delivery of data within a predefined time period. This can be particularly challenging in automotive and industrial automation, where real-time control is critical. Furthermore, Ethernet is susceptible to network congestion because it uses a shared medium for communication. This means that multiple devices can transmit data simultaneously, which in turn can cause collisions and delays that degrade network performance and make it difficult to transmit data in a timely manner.

A lot has been done by standardization bodies, academia, and industry experts in the last two decades to address these limitations and several changes and/or additions have been made to the Ethernet standards to include features for real-time communication (Nasrallah et al., 2019a). The latest set of *updated standards* in this direction is referred to as *Time-Sensitive Networking* (TSN) (IEEE-TSN, 2012). In this set, the IEEE 802.1Qbu (IEEE, 2016a) TSN sub-standard (now merged with the IEEE 802.1Q-2018 standard (IEEE, 2018a)) introduces a 1-*level frame preemption* scheme to the IEEE 802.1 networks. In contrast to the original Ethernet specification, this scheme allows the transmission of a frame to be temporarily suspended prior to its completion in order to expedite the transmission of a more urgent frame. This standard is closely related to the IEEE 802.3br (IEEE, 2016b) standard, which allows urgent time-critical frames to split non-critical frames transmitted over a physical link into smaller fragments (Lo Bello and Steiner, 2019). In particular, frames are mapped to two MAC sublayer service interfaces depending on their priority and timing requirements, namely (1) the "*Express MAC*" (eMAC) and (2) the "*Preemptable MAC*" (pMAC). Frames that map to the eMAC and pMAC interfaces are referred to as "express frames" and "preemptable frames", respectively. These two classes are managed as follows: (*i*) only express frames can preemptable frames; and (*ii*) two frames belonging to the same class cannot preempt each other. Figure 1.3 illustrates such a scenario.

In this figure, three frames (f_1, f_2, and f_3) are considered, and the priorities of the frames are set so that "the smaller the index of a frame, the higher its priority", i.e., f_1 gets the highest priority and f_3 the lowest. As in the previous example, an up arrow represents the arrival time of a frame and a down arrow represents the deadline. We assume that frames are transmitted according to a 1-level preemption scheme and are

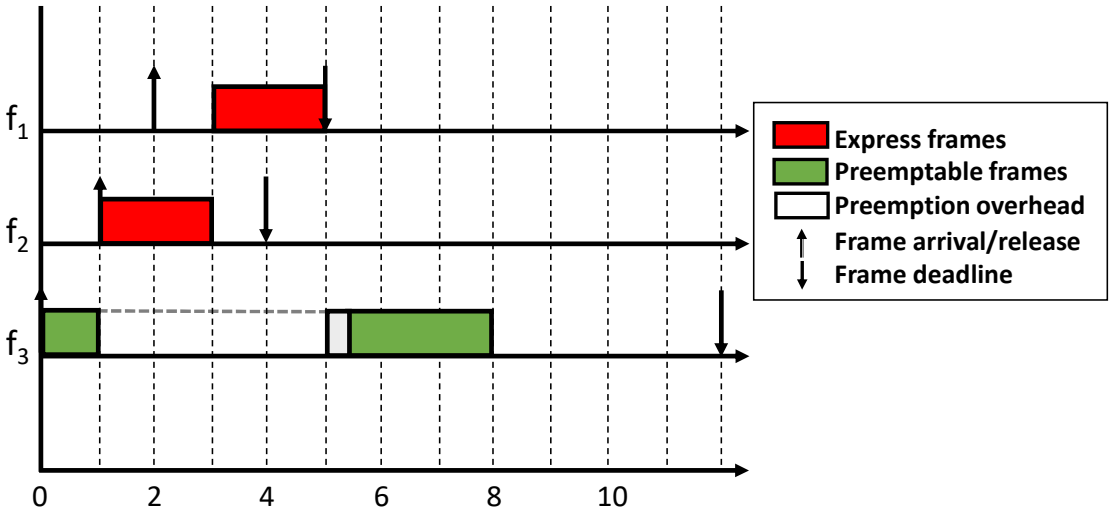


Figure 1.3: Illustration of the 1-level preemption scheme.

divided into two preemption classes: (1) the *highest preemption class (express)*, in which frames are represented by "red boxes"; and (2) the *lowest preemption class (preemptable)*, in which frames are represented by "green boxes". The light gray boxes represent the costs associated with the occurrence of a preemption. As can be seen, preemption favors prompt service of all express frames, but this comes at the cost of some overhead. The extra cost comes from the fact that fragments of split frames must form valid Ethernet frames. Namely, information such as the number of fragments (Fragment Count), an error correction code (CRC) to determine if all fragments were transmitted correctly, etc. must be added to the fragments of the preempted frame - here f_1 - so that the network nodes can send and receive them all correctly. The total overhead associated with the occurrence of each preemption is equivalent to the time required to transmit 24 bytes of data (i.e., $0.19\mu s$ and $1.9\mu s$, assuming an Ethernet with 1 Gb and 100 Mb speeds, respectively). Experimental studies show that this approach improves the performance of express frames (Thiele and Ernst, 2016a; Jia et al., 2013). In particular, Thiele and Ernst (Thiele and Ernst, 2016a) show that the performance of standard Ethernet with frame preemption in terms of worst-case delay guarantees is comparable to that of the so-called "Time-Aware Shaper" – a time-triggered gate control mechanism defined in the IEEE 802.1Qbv standard (IEEE, 2016c) for TSN frame transmission, making it an interesting alternative.

1.2 Motivation

Lo Bello and Steiner (2019); Gogolev and Bauer (2020); Ashjaei et al. (2021) pointed

out some limitations of the 1-level preemption scheme as specified in the standards. Most importantly, Lo Bello and Steiner (2019) and Gogolev and Bauer (2020) noted that this scheme is prone to performance degradation when the number of express frames is high. In this manuscript, we consider the scheme from a different perspective and point out that:

> The 1-level preemption scheme greatly improves the responsiveness of express frames but exhibits some serious limitations and poor performance when it comes to transmitting *preemptable frames with firm timing requirements*.

There are frames that cannot be classified as express frames, but have firm timing requirements. These frames should not be blocked for an excessive amount of time by lower priority frames, so as not to jeopardize the schedulability of these frames and thus the schedulability of the entire system. The current specification of the 1-level preemption scheme does not allow these frames to take advantage of the basic benefits of preemption. Recall that the TSN standards mandate that frames belonging to the same class cannot preempt each other. Figure 1.4 illustrates such a scenario.

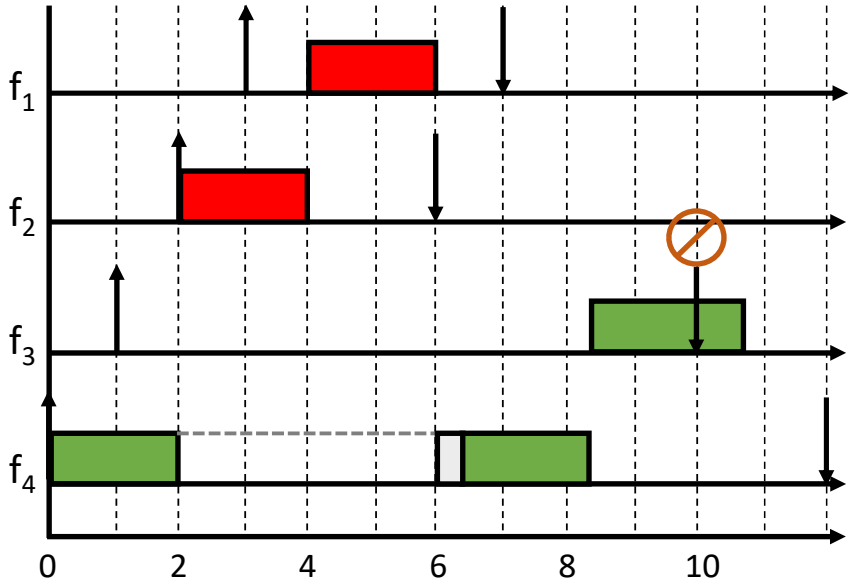


Figure 1.4: Limitation of the 1-level preemption scheme with f_3 as "preemptable".

In this figure, four frames (f_1, f_2, f_3, f_4) are considered and the priorities are set such that "the smaller the index of a frame, the higher its priority", i.e. f_1 gets the highest priority and f_4 the lowest. Express frames are represented by "red boxes" and preemptable frames by "green boxes". The light gray box represents the cost associated with the occurrence of a preemption. We are interested in the fate of frame f_3, which has a firm

deadline of 9 time units after its release at time $t = 1$. In this scenario, frame f_4 arrives first and begins its transmission. At time $t = 1$, frame f_3 (with a higher priority than f_4), arrives but cannot preempt f_4 because both belong to the same preemption class. At time $t = 2$, f_2 (which is express) arrives and preempts the transmission of f_4. At time $t = 3$, f_1 (which has a higher priority than f_2) arrives but cannot preempt f_2 since both belong to the same preemption class. After the express frames are completed, f_4 resumes its transmission despite f_3 being ready and pending. This is due to the actual specification of the 1-level preemption scheme and illustrates a priority inversion. Finally, due to this long blocking, f_3 eventually misses its deadline.

Intuitively, to circumvent this hurdle, one might want to move f_3 to the express class, since its timing requirement cannot be satisfied in the preemptable class. Figure 1.5 illustrates such a scenario.

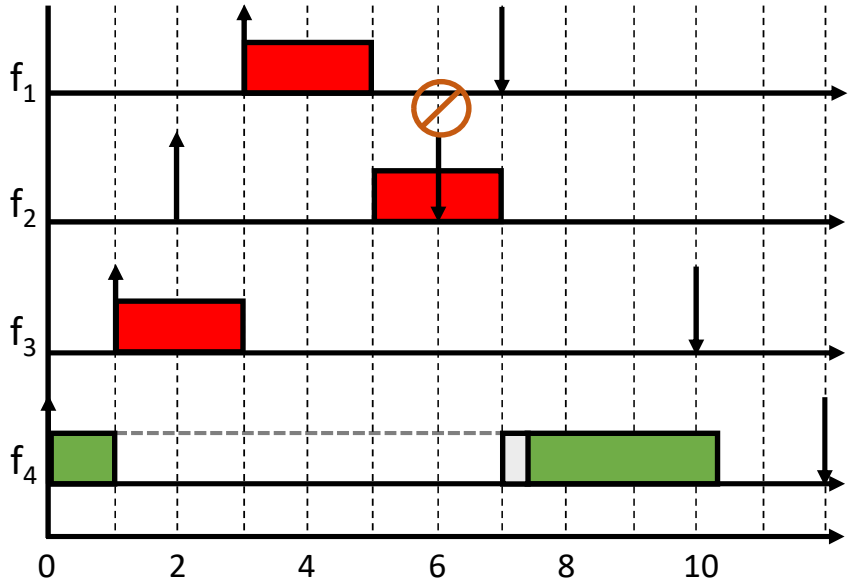


Figure 1.5: Limitation of the 1-level preemption scheme with f_3 as "express".

In this scenario, frame f_3 (which is now express) arrives and preempts the transmission of f_4. At time $t = 2$, frame f_2 arrives (with a higher priority than f_3), but cannot preempt f_3 since both belong to the same preemption class. At time $t = 3$, f_1 (with a higher priority than f_2) arrives and begins its transmission immediately after f_3. After the completion of f_1, f_2 starts its transmission. Consequently, f_2 misses its deadline due to the blocking by f_3 and the transmission of f_1.

The above challenges encourage us to think about relaxing the 1-level preemption constraint. Using a hypothetical 2-*level preemption scheme*, we re-examine the scenarios shown in Figures 1.4 and 1.5 and show the result in Figure 1.6.

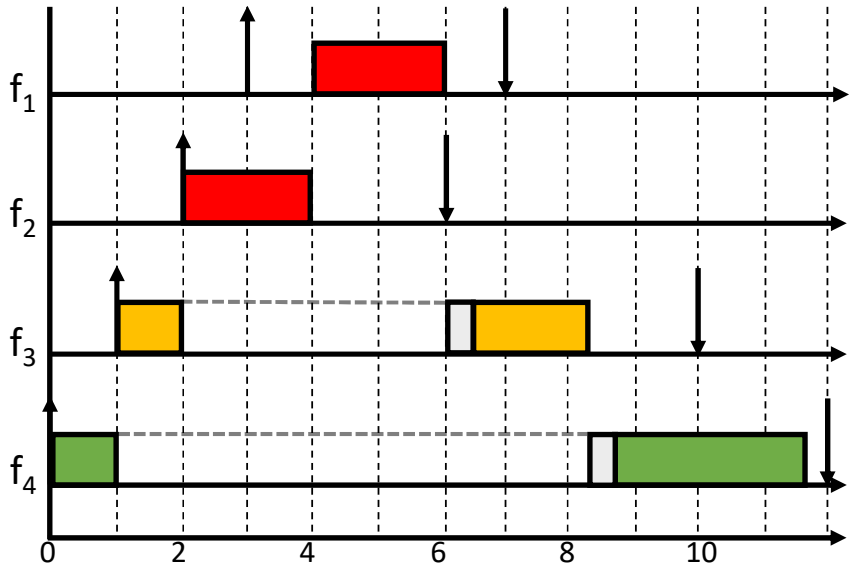


Figure 1.6: Frame transmissions under a 2-level preemption scheme.

In this figure, we now assume three possible preemption classes: (1) the *highest preemption class*, in which frames are represented by "red boxes"; (2) the *middle preemption class*, in which frames are represented by "yellow boxes"; and finally (3) the *lowest preemption class*, in which frames are represented by "green boxes". We assume that frames in a higher preemption class can preempt any frame in a lower preemption class and that frames in the same preemption class cannot preempt each other, similar to standards. As we can see from the figure, frame f_3 is better served because it is transmitted immediately after f_2 and f_1 are completed. This results in no priority inversion, and all frames are able to meet their timing requirements.

From this example, it is clear that preemptable frames with firm timing requirements (in the yellow-colored box) suffer from poor responsiveness under the 1-level preemption paradigm, but the situation already improves under a 2-level preemption paradigm. Since this type of frame is not uncommon in real-time applications, this provides a clear motivation to investigate not only a 2-level, but a *multi-level preemption scheme* in TSN to get the maximum benefit from this approach. Another notable limitation in the standards is the lack of configuration definitions to ensure optimal/efficient performance in terms of frame responsiveness. Eight Classes of Service (CoS) are defined in the specification of Ethernet frames (IEEE, 2014), but it is not clear which of them should be mapped to Express or Preemptable frames in the TSN specification. Last but not least, the TSN standards do not specify efficient routing techniques to optimize network performance.

1.3 Thesis statement and research questions

In this work, we propose solutions to the shortcomings of the 1-level preemption scheme specified in the TSN standards. In this framework, the thesis statement is as follows:

> *We postulate that multi-level preemption, when added to TSN, can mitigate several shortcomings of the frame preemption standard and unlock timeliness and resource utilization benefits of DRES.*

Based on this thesis, we investigate and promote the implementation of a **Multi-level Preemption Scheme**. With this new paradigm, we envision that all frames with strict and/or firm timing requirements receive a significant improvement in their performance in terms of responsiveness. This, in turn, allows us to improve both the Quality of Service (QoS) and the reliability of the overall system, and to increase the workload that can be accommodated in the underlying communication network. However, achieving this goal involves a number of challenging scientific problems, which are formulated in three research questions presented below.

(RQ_1). **Research Question 1:** At this stage, we are interested in investigating the operational feasibility of multi-level preemption within the specifications of the TSN standards. In particular, we are investigating how the scheme can be achieved with the least changes to the standard specifications while maintaining interoperability with other existing Ethernet/TSN transmission schemes. We are also interested in how the resource utilization overhead of the multi-level preemption scheme can be minimized and how this overhead compares to other TSN traffic control mechanisms from a qualitative perspective. Formally, the first research question is as follows.

> How feasible is a multi-level preemption scheme in TSN with the current specifications in the standards and what are the requirements for its efficient implementation?

(RQ_2). **Research Question 2:** Compliance with timing requirements is essential for any real-time communication medium. Therefore, formal timing analyzes must be performed to provide safe timing guarantees when a new feature and/or scheme is proposed. These analyzes not only provide real-time guarantees but also allow eval-

uation of the newly proposed feature and/or scheme from a quantitative perspective. The need to perform such analyzes for the proposed multi-level preemption scheme leads us to the second research question, which is formulated as follows.

> How does multi-level preemption affect the temporal behavior of time-critical data flows if we want to provide real-time guarantees?

(RQ_3). **Research Question 3:** The performance of any real-time transmission scheme depends heavily on the configuration chosen, and the multi-level preemption is no exception. Therefore, determining the appropriate configuration framework to achieve the best possible performance is a task that deserves our utmost attention. This fact leads us to the final research question, which is formulated as follows.

> How can a TSN network with a multi-level preemption scheme be configured to achieve efficient and/or optimal performance?

1.4 Research methodology

To validate the thesis, a deductive research methodology was used. In this method, a hypothesis is first developed based on the state-of-the-art. Then, a strategy is established to test the hypothesis using a series of observations to determine if it is supported by the data. The key characteristic of this approach is that the conclusion is necessarily true if the original principles or ideas are true. In the context of our research, the thesis is the main hypothesis. We arrive at this thesis after a thorough review of the Frame Preemption specification in the TSN standards and all related work in the literature on the subject. We identify three main research questions, namely RQ_1, RQ_2, and RQ_3, whose answers would allow us to confirm or reject the thesis.

RQ_1 is a qualitative assessment that involves a comprehensive analysis and comparison of different implementation profiles. In this framework, the qualitative evaluation criteria are (1) feasibility; (2) resource utilization; (3) ease of configuration; and finally (4) flexibility of a TSN network with multi-level preemption. Specifically, we conduct a comprehensive study of TSN standards, along with various other materials, including several presentations and review documents from the IEEE TSN Task Group (IEEE-TSN, 2012). In addition, collaboration with industry partners at Comcores ApS, Lyngby, Denmark is initiated to obtain data on the resource utilization profiles of various TSN traffic

control mechanisms. All this information is extensively analyzed and relevant conclusions and recommendations are derived.

In contrast to RQ_1, RQ_2 and RQ_3 are quantitative assessments. To answer these, we explore relevant scientific theories and tools. In addition, extensive discussions with thesis advisors and, in some cases, collaborators from the real-time communication research community were conducted. The first goal of these discussions is to develop a model that effectively captures the characteristics of a TSN network with multi-level preemption. Once this model is successfully developed, we identify the key metrics to be used in evaluating the proposed scheme. We then systematically and consistently conduct experiments to compare the new scheme to existing ones in the literature. The results of these experiments are analyzed in depth to draw unbiased conclusions about the effectiveness of the multi-level preemption scheme.

1.5 Overview of the contributions

In this section, we summarize the main findings of our research in relation to the research questions outlined in Section 1.3 and explain how our work contributes to the existing body of knowledge in the field of real-time communication. To this end, the section is organized under three headings, namely: (1) *feasibility and implementation cost*; (2) *model and analysis*; and finally, (3) *configuration and evaluation*.

1.5.1 Feasibility and implementation cost

This contribution addresses RQ_1. It examines the preemption mechanism in detail to determine the feasibility of a multi-level preemption scheme, and our analysis confirms its feasibility. First, we examine the Ethernet frame format to determine how different preemption classes can be encoded without radical changes to the standard specifications. We then examine the frame transmission and reception processes to understand how additional frame preemption classes would fit into the existing definitions. Special care was taken to avoid deadlock situations or transmission errors. We explain the changes required to achieve this goal and provide implementation recommendations to ensure frame integrity and interoperability. In this context, we also compare the hardware implementation costs of the multi-level preemption scheme with the time-triggered (TAS) and credit-based shaper (CBS) approaches. Here, the results show that the implementation cost of frame preemption, while significantly higher than that of CBS, is very much lower than that of the prevailing TAS. In particular, the results show significantly lower resource consumption for up to four preemption levels compared to TAS. Note that in this comparison

we do not consider the additional overheads of the time synchronization software required by TAS. The above contributions are presented in the following publications:

- **M. A. Ojewale**, P. Meumeu Yomsi and B. Nikolić, *Multi-Level Preemption in TSN: Feasibility and Requirements Analysis*, IEEE 23rd International Symposium on Real-Time Distributed Computing (ISORC), 2020, pp. 47-55,https://doi.org/10.1109/ISORC49007.2020.00017.

- A. Pruski, **M. A. Ojewale**, V. Gavrilut, P. Meumeu Yomsi, M. S. Berger and L. Almeida, *Implementation Cost Comparison of TSN Traffic Control Mechanisms*, 26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), 2021, pp. 01-08, https://doi.org/10.1109/ETFA45728.2021.9613463.

1.5.2 Model and analysis

This contribution answers RQ_2. It provides formal worst-case upper bounds on the delays experienced by each frame under the multi-level preemption scheme, using the so-called *Compositional Performance Analysis* (CPA) approach. For this purpose, four delay components are identified and summed, namely (1) the delay due to the transmission of a lower priority frame; (2) the delay due to the transmission of frames of the same priority; (3) the delay due to the transmission of higher priority frames; and finally (4) the delay due to the preemption overhead. Formal proofs are presented to establish numerical upper bounds for each of these components. We evaluate the safety of these upper bounds by comparing the numbers obtained with those from extensive simulations using NeSTiNg (Falk et al., 2019), a simulation model for TSN built on the OMNeT++ framework (OMNET++). We also evaluate the performance improvements over the 1-level preemption scheme using a realistic automotive use case. Our results show that the multi-level preemption scheme provides up to 53.07% improvement in terms of worst-case traversal time (WCTT) reduction for preemptable time-critical frames. The above contributions are presented in the following publications:

- **M. A. Ojewale**, P. Meumeu Yomsi, G. Nelissen, *On Multi-level Preemption in Ethernet*, Work-in-Progress (WiP) Session, 30th Euromicro Conference on Real-Time Systems (ECRTS), pp. 16-18, 2018.

- **M. A. Ojewale**, P. Meumeu Yomsi, B. Nikolić, *Worst-case traversal time analysis of TSN with multi-level preemption*, Journal of Systems Architecture (JSA), Volume 116, 2021, 102079, ISSN 1383-7621, https://doi.org/10.1016/j.sysarc.2021.102079.

1.5.3 Configuration and evaluation

This contribution answers RQ_3. It addresses the synthesis problem for multi-level frame preemption in TSN. Specifically, given a set of flows and network topology,we present a framework to (1) assign priorities to flows; (2) determine the appropriate level of preemption to enable; and finally (3) assign flows to preemption classes. We experimentally evaluate the performance of the proposed framework and our results show that the proposed scheme outperforms Deadline Monotonic Priority Ordering, which is known to dominate most other priority assignment schemes in the literature. Our approach ensures that only the required number of preemption levels are enabled since each additional preemption level is associated with significant hardware overheads that can increase switch manufacturing costs.

We also examine RQ_3 from the perspective of flow routing. Indeed, among others. Nayak et al. (2018), Singh (2017), and Gavrilut et al. (2017) highlighted the importance of routing in achieving low latency, predictability, and reduced architectural cost. In this work, we follow the same path and focus on the routing problem of TSN flows, since an inappropriate routing strategy can increase the number of transmission operations, resulting in additional delays. Moreover, the blocking time of flows in the network may increase if too many flows try to traverse the same path at the same time. We believe that a strategy that minimizes the number of transmission operations and the blocking time of each flow would help circumvent and/or mitigate these situations. Specifically, we proposed two routing heuristics, the algorithm *Load-Balanced, Dynamic, and Replication-aware Routing* (LB-DRR) and the algorithm *Congestion Recovering, Dynamic and Replication-aware Routing* (CR-DRR), to address the load-balancing and congestion issues in TSN. We evaluated the performance of the proposed schemes compared to the popular *Shortest Path* (SPA) and *Weighted Equal Cost Multi-Path* (wt-ECMP) routing algorithms and found an improvement of more than 70% and 20%, respectively. These improvements are observed in terms of the maximum load transmitted on an edge. Moreover, the proposed heuristics show high scalability in terms of increasing the number of flows. The above contributions are captured in the following publications:

- **M. A. Ojewale**, P. Meumeu Yomsi, and L. Almeida, *A Configuration Framework for Multi-level Preemption Schemes in Time Sensitive Networking*, 30th International Conference on Real-Time Networks and Systems (RTNS 2022). Association for Computing Machinery, New York, NY, USA, 219–229. https://doi.org/10.1145/3534879.3534891.

- **M. A. Ojewale** and P. Meumeu Yomsi, *Routing heuristics for load-balanced transmission in TSN-based networks*, SIGBED Rev. 16, 4 (December 2019), 20–25. https://doi.org/10.1145/3378408.3378411.

1.6 Significance of the contributions

The results presented in this dissertation have significant implications for the field of real-time communication. A recent survey of communication solutions in the industry shows that Ethernet is leading the race in several DRES domains (Akesson et al., 2020). In addition, studies have shown that the performance of standard Ethernet with frame preemption is comparable to that of the widely studied Time Aware Shaper (TAS), which is more complex and expensive to implement (Thiele and Ernst, 2016a; Lo Bello and Steiner, 2019; Gogolev and Bauer, 2020). Frame preemption is not only easier to implement and configure, but also allows more efficient use of network bandwidth, requires fewer hardware resources to implement, and is more flexible than TAS (Pruski et al., 2021; Gogolev and Bauer, 2020). However, several studies have pointed out that the major limitation of the current frame preemption specification in the standards is that it only allows one level of preemption (Lo Bello and Steiner, 2019; Gogolev and Bauer, 2020; Ashjaei et al., 2021).

> By addressing the main limitations of TSN frame preemption specifications, this work promotes standard Ethernet with multi-level frame preemption as a simpler and cheaper alternative communication solution that could lead to a paradigm shift in the development of future DRES.

1.7 Dissertation structure

The rest of this document is structured as follows.

- Chapter 2 introduces background concepts necessary for a smooth understanding of our contributions. First, real-time communication is introduced, and then an inexhaustive but representative set of protocols developed for real-time communication is discussed. Then Ethernet and TSN are examined in detail. Then the TSN frame preemption feature is discussed in detail and some of its limitations are mentioned. Finally, some related work relevant to this thesis is presented.

- Chapter 3 details the findings on the feasibility and implementation costs of multi-level preemption in TSN. It contains a detailed analysis of how the processes for

transmitting and receiving frames can be modified to support multi-level preemption. It also includes the requirements analysis results and some recommendations for implementation. In addition, the chapter quantifies the hardware implementation cost for multi-level preemption on an FPGA platform and compares it to the implementation cost of other TSN traffic control mechanisms.

- Chapter 4 discusses the framework chosen for modeling and analysis in this thesis. This includes the network, traffic, and configuration models and assumptions, as well as an explanation of the CPA framework. The chapter also covers a detailed WCTT analysis of traffic flows under a multi-level preemption scheme using the CPA approach and presents the configuration framework for this scheme.

- Chapter 5 contains the extensive quantitative evaluations of both the WCTT analysis and the configuration framework presented in the previous chapter. Using synthetic and real-world use cases, we comprehensively evaluate the performance of the multi-level preemption scheme.

- Chapter 6 presents routing heuristics to improve the responsiveness and reliability of frame transmission in TSN. In particular, two routing heuristics, referred to as LB-DRR and CR-DRR, have been proposed to solve the congestion avoidance and congestion recovery problems, respectively.

- Chapter 7 contains a summary of the results of the thesis, the validation of the thesis, and the relevant conclusions. Finally, this chapter discusses the limitations of the thesis and some directions for future work.

Chapter 2

Background

In this chapter, we introduce background concepts necessary for a smooth understanding of our contributions. In particular, we discuss the fundamentals of real-time communication, Ethernet technologies, TSN, and Frame Preemption in the context of DRES. In this regard, the rest of the chapter is organized as follows. First, we introduce real-time communication in Section 2.1. In Section 2.2, we describe a representative, but not exhaustive, selection of established real-time communication protocols from the literature. We then take a closer look at Ethernet and TSN, respectively, in Section 2.3 and Section 2.4. Then, in Section 2.5, we discuss the TSN frame preemption feature as currently specified in the standards and recall its main limitations discussed in the previous chapter. Finally, in Section 2.6, we present a number of related works that are relevant to the challenges addressed in this dissertation.

2.1 Real-time communication

The proper operation of DRES depends not only on the timely execution of computational tasks, but also on the timely communication between the embedded devices that make up the system (Malcolm and Zhao, 1992) For traditional communication network applications such as electronic mail, file transfer, and remote desktop connections, the typical performance metrics of interest are: throughput, delay, and average messages/packets delivered (Aras et al., 1994). For these types of applications, it is usually desirable to achieve low latency. Much of the complexity of network protocols arises from the need for lossless communication, and there are no specific timing constraints on messages. In contrast, DRES consists of multiple embedded real-time devices that must communicate within precise timing constraints. The distinguishing feature of real-time communication is that the value of the communication also depends on the time at which the messages

are successfully delivered to the receiver (Aras et al., 1994; Cottet et al., 2004). Here, each message is assigned a so-called *deadline* –the latest desired delivery time–and the value of the message is greatly diminished if it is delivered after its deadline. In some real-time applications, messages are even considered ”perishable” and unusable if they are not delivered on time. In these applications, missing a deadline could lead to serious and/or catastrophic consequences, such as loss of life, injury to people, and/or financial loss. Figure 2.1 illustrates a real-world use case with real-time communication, where an autonomous vehicle (the black car) crosses an intersection and detects an obstacle (the gray car in the red rectangle).

Figure 2.1: Obstacle detection in an autonomous vehicle. Source: Zendrive

In this figure, the autonomous vehicle’s detection sensor must not only correctly inform the braking system that an obstacle has been detected, but it must also transmit this information quickly enough for the braking system to make a decision in time to avoid a collision. If this information reaches the braking system only after a collision has occurred, it is useless. Therefore, real-time communication must be reliable and predictable, i.e., it must be ensured that the time required to relay the message does not exceed a predefined value that is less than or equal to the deadline. Other use cases include the Global Positioning System (GPS), multimedia systems, and surveillance systems, to name just a few examples.

In the dissertation, we focus on real-time communication in the context of DRES.

2.2 Real-time communication protocols

Historically, real-time communication was implemented using point-to-point (P2P) connections between components (nodes) that need to communicate with each other (Davis et al., 2007). However, the number of connections (wires) required for a fully connected network increases very rapidly. In fact, the number of wires required for a fully connected P2P network with n nodes is $\frac{n\times(n-1)}{2}$ – that is, this number grows polynomially with the number of nodes. A fully connected network with only 5 nodes requires up to $\frac{5\times(5-1)}{2} = 10$ wires, as shown in Figure 2.2.

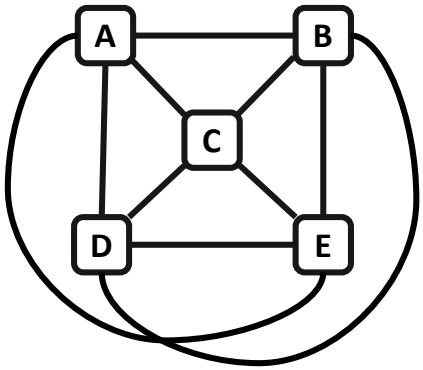


Figure 2.2: Illustration of a fully connected P2P network with 5 nodes.

With the ever-increasing number of interacting nodes in DRES, managing the number of wires becomes very difficult. To illustrate this assertion, suffice it to mention that an autonomous car today consists of over 100 Electronic Control Units (ECUs). This would mean at least 4950 wires to achieve a fully connected underlying network. To get around this hurdle, the so-called "bus" communication system was introduced (Gustavson, 1984). Unlike P2P communication networks, this is a shared communication path between multiple nodes. Here, a node must first be granted exclusive access to the shared communication path before it can send data. Once this access has been granted, the node can communicate. In one of the first access methods, namely master-slave, the sending node (called the "master") writes its message to the bus, and this is accessible to all other nodes (called the "slaves"). The message contains an identifier, the addresses of the sender and receiver nodes, the data to be transmitted, and whether the communication event is a *read* or a *write* (Gustavson, 1984). The term "read" means that the master requests some information from the receiving node(s), while the term "write" means that the master shares some updates with the receiving node(s). After the message is received by the slaves, only the node(s) with the matching address(es) will process the message

and perform the required actions. The other nodes will ignore and discard it. Access to the shared communication path is controlled by the so-called Medium Access Control (MAC) protocol, which ensures that only one node can access the bus at a time. Note that this MAC varies from one bus implementation to another. Figure 2.3 shows an example where the previous P2P network in Figure 2.2 is converted to a bus-based network.

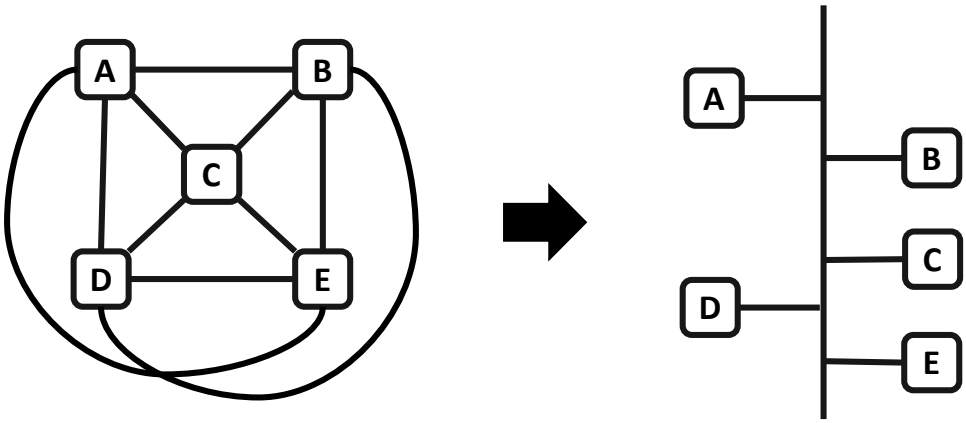


Figure 2.3: Bus communication with 5 nodes

In the remainder of this section, we briefly describe some of the most prominent shared medium real-time protocols in the literature.

2.2.1 Fibre Distributed Data Interface (FDDI)

FDDI is a shared network with a ring topology (Cottet et al., 2004) in which the access control is performed using a so-called "token". The token is passed from one node to the next, in the order in which it is located in the physical ring network. A node can only transmit if it has the token. Conversely, if a node has nothing to transmit, it forwards the token immediately to the next node. However, to avoid the starvation of other nodes that would result from a node holding the token for an unbounded period of time, a parameter called Target Token Rotation Time (TTRT) is used. The TTRT specifies approximately the maximum time the token may take to traverse the entire ring. When a node receives the token, it measures the time the token took to circulate through the whole ring (called the Real Token Rotation Time) and subtracts this from the TTRT. This difference is called the Token Holding Time. If it is positive, the node is allowed to transmit during this time. If it is zero or negative the node must forward the token immediately. Consequently, the maximum time a node can hold the token and transmit is bounded, making token-based communication predictable. Figure 2.4 shows a simplified architecture of an FDDI network with 5 nodes.

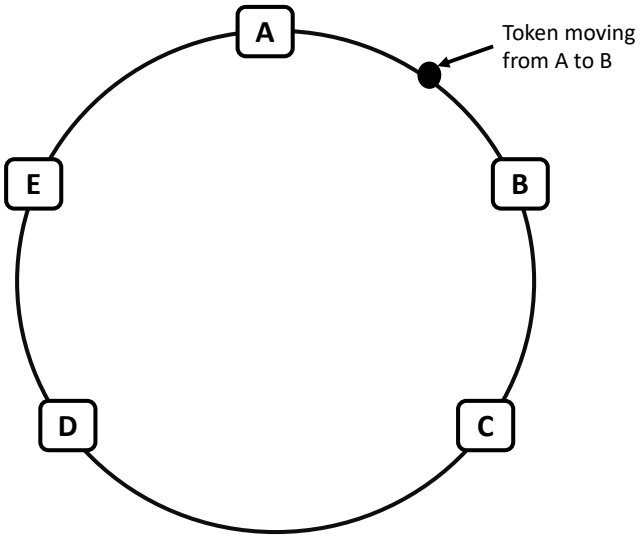


Figure 2.4: Simplified architecture of an FDDI network with 5 nodes.

The *token-bus* is a variant of FDDI in which the shared medium is a bus instead of a ring. Here, the logical successor of a node is not necessarily the same as its physical successor, which gives the MAC protocol additional flexibility.

2.2.2 Controller Area Network (CAN)

CAN is also a bus-based communication technology and is often referred to as a "CAN bus". Its shared medium consists of a twisted pair of wires ("CAN high" and "CAN low") to which all nodes are connected in parallel. The extremities of the CAN bus must be connected to terminating resistors of 120Ω (nominal) to reduce signal reflections (Richards, 2002). Data transmission on a CAN bus is accomplished by a differential voltage signal between the CAN high and CAN low wires. When a device transmits logical 1, it applies a higher voltage to the CAN high wire and a lower voltage to the CAN low wire. Conversely, when a logical 0 is transmitted, it applies a lower voltage to the CAN high wire and a higher voltage to the CAN low wire. This creates a differential voltage between the two wires that is used to transmit the data. Other devices in the CAN network listen for this voltage difference and can decode the transmitted message. The transmission in differential voltage mode increases the resilience to electromagnetic interference. Figure 2.5 shows a simplified architecture of a CAN network with 5 nodes.

CAN uses a bit modulation called Non-Return-to-Zero that corresponds to keeping the voltage levels fixed during the whole bit time. The bus drivers of each node implement a so-called wire-AND function that grants the property of dominance. The logical 0 is called dominant while the logical 1 is called recessive. A single node transmitting a 0

forces the bus level to 0. Conversely, the bus level will be at 1 only when all nodes are transmitting 1 (Richards, 2002)

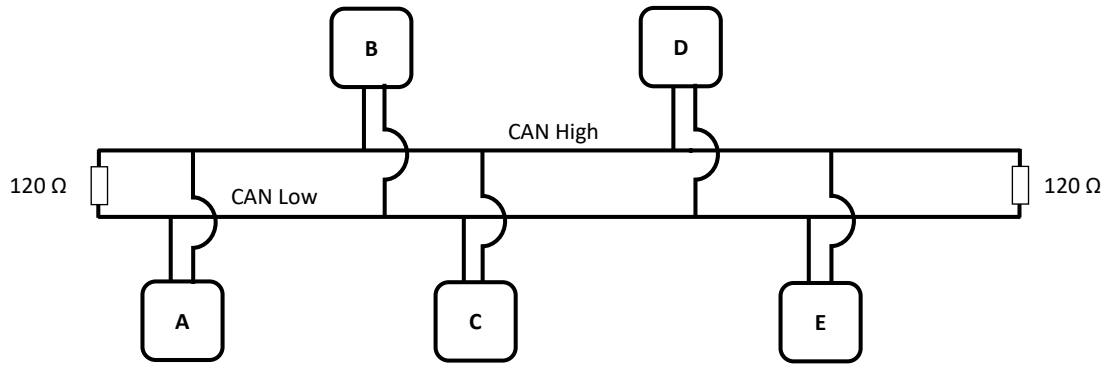


Figure 2.5: Simplified architecture of a CAN network with 5 nodes.

Nodes access the CAN bus randomly and event-driven. If two nodes attempt to occupy the bus simultaneously, access is granted by non-destructive, bitwise arbitration, i.e., the node that wins the arbitration simply proceeds with the message without the message being destroyed or corrupted by another node. The identifier of a message (ID) is also its priority, and the lower the binary identifier of a message, the higher its priority. A node that wins the arbitration puts its data (called a "signal") on the bus, and all other nodes in the network have access to it. However, only the receiving node(s) receive(s) this message from the bus, while all other nodes simply ignore it. This protocol (Szydlowski, 1992) was introduced by Robert Bosch GmbH in the early 1990s and was primarily intended for the automotive sector to meet speed and bandwidth requirements.

Regarding the advantages and disadvantages of this protocol, it should be noted that CAN is very well-suited for applications with a large number of short messages. In addition, it is particularly well-suited when data is needed from more than one location due to the broadcast nature of the communication. However, it also has some serious limitations. First, the length of the CAN bus is limited. The maximum bus length at a bit rate of 10 kbit/s is 1 km, and the maximum length at 1 Mbit/s is 40 m.

The data rate during the arbitration phase cannot exceed 1 Mbit/s, even in more recent versions that allow higher speeds during the transmission of the payload of the message (e.g., CAN-FD). Due to its broadcast nature, the CAN protocol does not efficiently utilize the communication medium. Furthermore, there is no robust flow control mechanism other than priority arbitration.

2.2.3 Local Interconnect Network (LIN)

At the turn of the millennium, the LIN protocol emerged as a complementary technology to CAN (Specks and Rajnák, 2000), providing low-cost, low-speed connectivity for in-vehicle networks. LIN is a serial, broadcast, one-wire (as opposed to CAN's two-wire) interface and is typically implemented as a sub-bus of a CAN network. It allows automotive manufacturers to reduce costs by moving slow, non-safety-critical functions from a two-wire bus CAN to a single-wire. LIN is a master-slave network in which a master coordinates communication between up to 16 slave devices on the network.

2.2.4 FlexRay

FlexRay ("Flexible Ray") is another bus-based communication protocol that also complements CAN in automotive networks (Pop et al., 2006) by providing higher speed (up to 10 Mbps) and more reliable communication on the sub-bus of a CAN network. FlexRay's higher speed means that its maximum bus length is significantly less than that of CAN. FlexRay is also significantly more expensive than CAN. It supports timed communication by performing cycle-based transmission. Specifically, each cycle is divided into two parts: the static segment and the dynamic segment. The static segment is divided into slices for individual communication types and provides stronger determinism than CAN, while the dynamic segment provides event-driven behavior.

2.2.5 Ethernet

The Ethernet communication protocol was developed by Robert Metcalfe at Xerox PARC between 1973 and 1974 (Shoch, 1981). The name is based on the word "ether" – an archaic word coined in the nineteenth century to describe the invisible medium through which light supposedly travels (Huang et al., 1998). Ethernet provides P2P communication via a simple passive medium (coaxial, twisted pair, or fiber optic cable) in which nodes communicate by sending messages to each other in the form of data packets. Each node is assigned a unique address and link-level communication is established using the addresses of the sender (source) and the receiver (destination). The receiver accepts only the packets addressed to it and ignores the rest. In 1980, the speed of Ethernet was already 10 Mbit/s (Shoch, 1981). Similar to bus-based technologies, Ethernet was originally designed to use a common P2P medium for communication, creating a single collision domain. To avoid collisions, a fully distributed link access mechanism called *Carrier Sense Multiple Access with Collision Detection* (CSMA/CD) is used (Shoch, 1981). Specifically, a node can transmit its message only when the shared medium is not busy. Since

all nodes are waiting for the medium to become free, it is possible for two or more nodes to detect the idle state at the same time and start transmitting. This eventually leads to a collision. In this case, the affected transmissions are aborted and the nodes wait a random amount of time before attempting to retransmit the messages. This process is repeated until the message is successfully transmitted, i.e., without collision.

The collision and retransmission of messages lead to low efficiency in bandwidth usage of Ethernet.

2.3 Ethernet for real-time communication

Although Ethernet predates CAN by nearly two decades and offers much more speed and bandwidth, many did not consider Ethernet as a solution for DRES until much later. This is because the older Ethernet standards were originally intended for non-real-time applications and have several limitations that make it challenging to use in DRES (Nasrallah et al., 2019a). As discussed in Section 1.1, one of these limitations is that it is unpredictable, i.e., there is no guarantee that data will be delivered within a predefined time period. Another limitation is its susceptibility to network congestion. Ethernet uses a shared medium for communication, which means that multiple devices can transmit data over the same network at the same time, which can cause collisions and delays that degrade network performance and make it difficult to transmit data in real-time.

Over the years, several changes have been made to these standards to address these shortcomings. For example, the IEEE 802.1p Task Group specified the so-called *Class of Service (CoS)* mechanism to speed up the transmission of some frames (IEEE, 2014). In particular, this mechanism introduced the notion of *priority* for Ethernet frames, where the CoS of a frame indicates its priority. With this mechanism, priorities can be assigned to FIFO queues and frames are stored in these queues based on their CoS classes. Eight CoS classes (IEEE, 2014) are defined, and frames are transmitted according to their CoS - the frames with the highest CoS first. Frames with the same CoS are stored in the same output queues and transmitted on a FIFO basis.

In the following sections, we introduce prominent protocols/features that have been introduced in the literature to mitigate the limitations of Ethernet for real-time communication.

2.3.1 Switched Ethernet

The two problems, i.e., collision and retransmission of messages (also referred to as "frames"), were the main reasons for the introduction of Ethernet switches. Unlike classical Ethernet, where the medium is shared by the nodes, Switched Ethernet gives each node its own connection to a switch through its various ports. The switch then connects the node to one or more other switches and nodes in the network. With Switched Ethernet, collisions do not occur in the medium because of the dedicated connection between each node and the switch(es). However, collisions can still occur on a destination port if it receives frames from more than one port at a time. In a switch, each port has its own individual collision domain and resolves them individually. Figure 2.6 shows a simplified architecture of a Switched Ethernet network with 5 nodes.

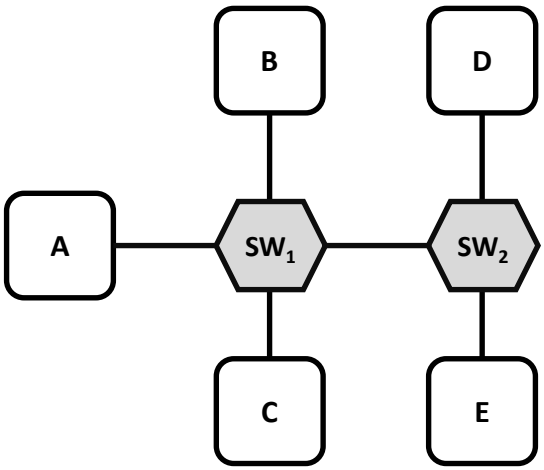


Figure 2.6: A simplified architecture of a switched Ethernet network with 5 nodes.

Each switch significantly mitigates the impact of the CSMA/CD arbitration mechanism on bandwidth utilization and makes the network less unpredictable. The internal architecture of a basic Ethernet switch is shown in Figure 2.7, where the number of input and output ports are equal.

Frames enter the switch through the input ports and leave through the output ports. The Frames are first received in a buffer and, after complete and error-free reception, are sent to the switch fabric for forwarding. The switch fabric uses the destination address and the predefined transmission rules to determine the output ports to which the received Frames should be forwarded. The output ports are equipped with multiple FIFO queues that store frames waiting to be forwarded. In the remainder of this dissertation, the term "Ethernet" will be used simply to refer to Switched Ethernet unless otherwise noted.

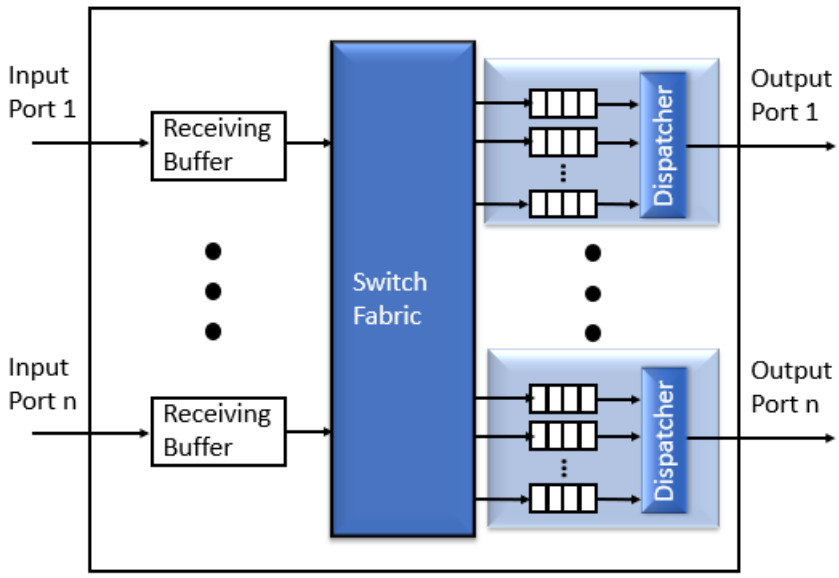


Figure 2.7: Illustration of the internal Architecture of a basic Ethernet Switch

2.3.2 EtherCAT

EtherCAT, or "*Ethernet for Control Automation Technology*", is a master-slave protocol developed by Berkhoff in 2003 and later standardized by the International Electronical Commission (IEC, 2007). The protocol was originally developed with industrial control systems in mind. In a typical EtherCAT network, there is a single master node and multiple slave nodes. In contrast to Ethernet, where frames are sent from sender nodes to destination nodes, EtherCAT nodes communicate via so-called "telegram" messages. Specifically, telegrams are generated by the master nodes and sent through the network, passing through all slave nodes and being sent back to the master. The slave nodes read the data addressed to them as the telegram traverses the node, and process the data "on the fly". Note that these nodes can also insert data into the telegram as it passes through them. EtherCAT supports data rates of up to 1 Gbit/s and can be used over both copper and fiber optic cables. It is developed and maintained by the EtherCAT Technology Group (ETG), a global organization that promotes the use of EtherCAT and other industrial Ethernet technologies. The ETG works closely with leading industrial automation companies and organizations to ensure that EtherCAT meets market needs and remains at the forefront of industrial Ethernet technology.

2.3.3 PROFINET

PROFINET, a portmanteau for Process Field Net, is a technical communication standard based on Ethernet (Feld, 2004). It is part of the IEC 61158-2 standard (Winkel, 2006) and is widely used in manufacturing and other industrial environments. PROFINET enables real-time communication and control of automation devices such as sensors, motors, and controllers. It supports (1) a wide range of data rates, from 100 Mbit/s to 1 Gbit/s; and (2) four real-time traffic classes, namely: *RT class UDP* (RT UDP), *RT class* 1 (RT 1), *RT class* 2 (RT 2), and *RT class* 3 (RT 3). RT 3 has the highest priority, followed by RT 2, then RT 1, and finally RT UDP the lowest. In PROFINET, communication is cyclic and each cycle is divided into communication intervals. These intervals are reserved specifically for each of the classes. The nodes use a clock synchronization mechanism to ensure that the cycles are synchronized. All other traffic is blocked when a scheduled slot occurs. The protocol also supports a non-real-time class (NRT), which has a lower priority than the real-time classes and is transmitted largely based on standard Ethernet specifications.

2.3.4 AFDX

Airbus developed and trademarked what it calls *Avionics Full-Duplex Switched Ethernet* (AFDX) in 2006 to improve the timing performance and reliability of standard Ethernet. As the name implies, AFDX targets real-time communication in avionics. Here, each node in the network is connected to a switch via a full-duplex link, meaning frames can move in opposite directions on a physical medium simultaneously. In this way, collisions can no longer occur on the physical medium, and CSMA/CD is no longer required. Full-duplex bidirectional transmission is achieved by using twisted pairs or fiber optic cables. While transmission links are collision-free, AFDX shifts the collision problem to the switching layer, where different frames compete for switching resources (e.g., input buffers, output queues, access to the transmission link, etc.). This can lead to congestion and frame loss at the switch's output ports if, at any given time, too many traffic arrives at a port than it can buffer or transmit. In addition, these congestion and frame losses negatively impact timing performance and reliability. To mitigate this, the protocol defines a virtual circuit - that is, a logical communication channel between a source node and one or more destination nodes. Each Virtual Link (VL) is associated with a so-called *Bandwidth Allocation Gap* (BAG), which defines the minimum time interval between successive frames on the VL. To enforce the BAG, AFDX defines a so-called "*shaper*" that limits the traffic rate at the source node to ensure deterministic communication behavior. The traffic resulting from this "shaper" became known as *Rate-Constrained (RC)* traffic.

2.3.5 TTEthernet

The concept of *Time-Triggered Ethernet* first appeared in the work of Kopetz et al. (2005) and TTEthernet is an industrial development that builds on this work (Steiner et al., 2009). Here, messages are divided into three traffic classes: (1) Time-Triggered (TT) traffic; (2) Rate-Constrained (RC) traffic (as discussed in AFDX, see Section 2.3.4); and finally (3) Best-Effort (BE) traffic. As the name implies, the main communication mode in TTEthernet is the *time-triggered* transmission. Specifically, TT traffic enjoys accelerated transmission and is transmitted based on a strict, pre-calculated offline schedule. When a time slot in the schedule is free of TT traffic, the available bandwidth is used for other traffic classes. In addition, TTEthernet requires a common time notion among the communicating nodes and thus network-wide clock synchronization. TTEthernet can provide deterministic behavior and timing guarantees for TT and RC traffic. However, the use of static offline schedules severely limits its configuration flexibility. Moreover, scheduling TT frames is equivalent to a bin-packing problem known as NP-hard (Nayak et al., 2018).

> The time synchronization requirement is not necessary (nor applicable) for some real-time systems, especially in use-cases that require dynamic behavior and/or unsynchronized terminals.

2.3.6 Audio Video Bridging (AVB)

In 2005 the IEEE formed the *Audio Video Bridging* (AVB) Task Group to develop an Ethernet profile for the automotive industry (IEEE, 2011a). The Task Group was particularly interested in supporting the transmission of time-sensitive media frames over automotive Ethernet networks. To accomplish this, AVB introduces three new standards: (1) *clock synchronization* (IEEE 802.1AS (IEEE, 2011b)); (2) *bandwidth reservation* (IEEE 802.1Qat (IEEE, 2010)); and (3) *traffic shaping* (IEEE 802.1Qav (IEEE, 2010)). It defines two Stream Reservation classes: Class *A* and Class *B* (where Class *A* has a higher priority than Class *B*) and only frames with real-time requirements are assigned to these classes. The transmission of frames is governed by the so-called *credit-based shaper* (CBS) algorithm (explained in detail in Section 2.4.4.2), which guarantees fair access for time-sensitive frames in the network and ensures that frames are delivered with minimal delay.

Although the AVB profile brings significant performance improvements to Ethernet, the standards can only guarantee a latency limit of 2 ms or less over 7 hops. Some emerging automotive applications require as low as 500μs – 1ms over several hops (IEEE, 2021).

2.3.7 Time-Sensitive Networking (TSN)

In 2012, the IEEE renamed the AVB Task Group the *Time-Sensitive Networking* (TSN) Task Group to reflect the broadened scope of its work, which is to "provide the specifications that enable time-synchronized, low-latency streaming services over IEEE 802 networks." (IEEE-TSN, 2012). In other words, the TSN Task Group is no longer focused only on the automotive domain but is expected to improve AVB standards and also extend the Ethernet Virtual Local Area Network (VLAN) protocol to meet the needs of time-sensitive applications in multiple domains.

In response, the TSN Task Group published (and is still publishing) a set of standards to introduce features that enable Ethernet to meet the stringent timing, bandwidth, and Quality of Service (QoS) requirements for time-sensitive data in emerging DRES. These standards will be referred to hereafter as "TSN standards" or simply as "TSN". They deal with network transmission at the flow (also referred to as "stream") level. Each *flow* is defined as an end-to-end unicast or multicast communication between two or more nodes. One node is referred to as the sender and the other(s) as the receiver(s). By this definition, each flow consists of a potentially infinite collection of frames transmitted from one node, e.g., A, to another node, e.g., B, separated by a minimum inter-arrival time between frames, also called *period*.

TSN is designed to support low-latency flow transmission with high reliability and precise synchronization. This makes it ideal for applications where real-time data is critical, including DRES.

2.4 TSN features in details

TSN uses a combination of several features introduced in different standards to achieve real-time communication. In this section, we discuss these features in more detail. Recall that TSN standards are motivated by the shortcomings of traditional Ethernet for real-time communication, including: (1) the lack of network-wide time synchronization; (2) the lack of network resource reservation and enforcement mechanisms; (3) the lack

of flow/traffic control mechanisms; and (4) the lack of appropriate flow integrity mechanisms. In the following subsections, we discuss TSN standardization efforts that address these shortcomings. For a structured approach to this discussion, we adopt a classification similar to Nasrallah et al. (2019a).

2.4.1 Network-wide clock synchronization

The IEEE 802.1AS (IEEE, 2011b) standard uses a profile of the IEEE precision time protocol (PTP) 1588-200 (IEEE, 2008) called generic PTP (gPTP) (Johas Teener and Garner, 2008) to synchronize clocks in a distributed network with a master-slave architecture. Specifically, it first uses what is called the *Best Master Clock Algorithm* (BCMA) to select the so-called GrandMaster (GM) - i.e., the node in the network with the most accurate clock[1]. Then a hierarchical spanning tree is created to communicate this time value across the network. In this process, each node serves as a master for all its child nodes. After receiving a clock value, each child node calculates its current time using a peer-path delay mechanism, i.e., an end-to-end communication delay between neighboring links with respect to the GM

From this description, the direct conclusion is that the synchronization of the network clock depends heavily on the availability and correctness of GM. In other words, the network is not immune to the failure of GM. To mitigate this limitation, a revision of the IEEE 802.1AS standard (IEEE, 2019a) has been introduced to improve *fault tolerance*. The revision supports the presence of multiple timing references in the network. This would allow the network to switch to a different time domain if one GM fails. Another limitation of IEEE 802.1AS is the message transmission overhead associated with time synchronization. More centralized approaches have been proposed in the literature. However, such an approach would introduce a single point of failure into the network.

> We believe that further studies are needed to find a good compromise between distributed and centralized approaches to leverage the strengths of both.

2.4.2 Resource reservation

The IEEE 802.1Qat (IEEE, 2010) defines admission control and resource reservation mechanisms for full-duplex switched Ethernet networks. In this standard, the so-called "*Stream Reservation Protocol (SRP)*" is introduced to ensure that both latency and bandwidth requirements can be met for each flow before it is admitted to the network. To this

[1]The GM time source is International Atomic Time (TIA) (Nasrallah et al., 2019a)

end, three other protocols (all defined in the IEEE 802.1Qat standard) are coordinated and used, namely: (1) the Multiple MAC Registration Protocol (MMRP); (2) the Multiple Stream Registration Protocol (MSRP); and finally (3) the Multiple VLAN registration Protocol (MVRP) (IEEE, 2010, 2014; Nasrallah et al., 2019a). As for the actual implementation of the SRP scheme, both resource requests and reservation requests of each flow are forwarded through the MMRP, while its VLAN membership in the network is registered with the MVRP. The MSRP performs the actual resource reservation throughout the network by communicating with all nodes on the path of the flow in a distributed manner. In short, the QoS requirement of each data flow is satisfied by: (1) announcing the data flow in the network; (2) registering the data flow path; (3) calculating its worst-case turnaround time; (4) establishing a communication domain between nodes where the reservation is enforced; and finally (5) reserving bandwidth for the data flow (Nasrallah et al., 2019a)

Since the Multiple Registration Protocol (MRP) is decentralized, the database of traffic information at each node grows with the number of traffic flows. As a result, the traffic information to be exchanged between nodes becomes very large. To overcome this limitation, the IEEE TSN working group has developed a new protocol called *Link-Local Reservation Protocol* (LLRP) (Finn, 2019). Unlike the MRP, this approach operates in a centralized manner. Such an approach reduces the communication overheads in resource management, but the configuration node may become a single point of failure. Consequently, this further emphasizes that the trade-off between centralized and distributed approaches remains an open problem that requires further research efforts.

2.4.3 Flow integrity

The IEEE 802.1CB (IEEE, 2017) standard defines a mechanism for transmitting multiple copies of a frame. This is to ensure that a frame can be delivered even if errors such as link failures, bitflips, or congestion losses occur on one of its paths. In the specification of this standard, if multiple copies of a frame arrive at a node (regardless of whether it is an intermediate node or the destination node), only one copy is received and all others are discarded. To achieve a high level of reliability, it is desirable that each copy of a replicated frame traverse its own path from the sender node to the receiver node. However, achieving this is not trivial. There are recent research efforts in this direction (Ojewale and Yomsi, 2020; Singh, 2017). Moreover, some user applications, such as “lock” – “ unlock” commands, sensor inputs, etc., require frames to be delivered in a specific order when replicas are present. In addition to 802.1CB, the IEEE 802.1Qca (IEEE, 2016d) standard allows the path of each data flow to be calculated in advance. In this process,

calculating the best routes for each data flow is not trivial, and therefore there are research efforts that address the selection of routing paths for TSN data flows.

> Finally, the IEEE 802.1Qci (IEEE, 2017a) standard allows data flows to be aggregated, processed, filtered, and queued based on their unique identifiers (Nasrallah et al., 2019a). This is very useful for security purposes, as unidentified flows can be easily discarded.

2.4.4 Flow control

TSN flow control mechanisms determine how frames belonging to a particular traffic class are handled at the output ports of TSN-enabled switches. These mechanisms include: (1) Time-Aware Shaper (TAS); (2) Credit-Based Shaper (CBS); (3) Cyclic Queue Forwarding (CQF); (4) Asynchronous Traffic Shaper (ATS); and finally the (5) Frame Preemption (FP). We describe these mechanisms in detail below.

2.4.4.1 Time-Aware Shaper (TAS)

To ensure low delay and low jitter for time-sensitive traffic, the IEEE 802.1Qbv (IEEE, 2016c) standard defines a *timed gate control mechanism*, also known as a *Time-Aware Shaper (TAS)*. TAS prescribes *independent time windows* by opening and closing the gate associated with each queue of a switch output port. Here, each queue has a guaranteed transmission slot in a cyclic and precomputed dispatch schedule called *Gate Control List (GCL)*. A frame in a queue can only be transmitted if the gate of the queue is opened at the time specified in the GCL configuration. Specifically, TAS follows the TDMA (Time Division Multiple Access) paradigm, which means that the transmission time is divided into time slots or “windows”. The time-sensitive frames are transmitted in the so-called “scheduled traffic window” or “protected window” and the non-scheduled traffic is transmitted in the “common slots”, i.e., slots for which no time-sensitive frame is scheduled. Each scheduled traffic window is preceded by a so-called “guard band"’ to prevent unscheduled traffic with a lower priority from blocking the transmission of scheduled traffic. The guard band is a period of time between the gate-close operation of non-time-sensitive queues and the start of the scheduled traffic window. The length of the guard band corresponds to the time required to transmit the largest possible non-time-sensitive frame in the network. Figure 2.8 shows an example of frame transmission with TAS.

In this figure, we consider two flows (f_1 and f_2) for transmission. We assume that f_1 is a scheduled time-sensitive flow, whereas f_2 is not time-sensitive. The frames of f_1 are

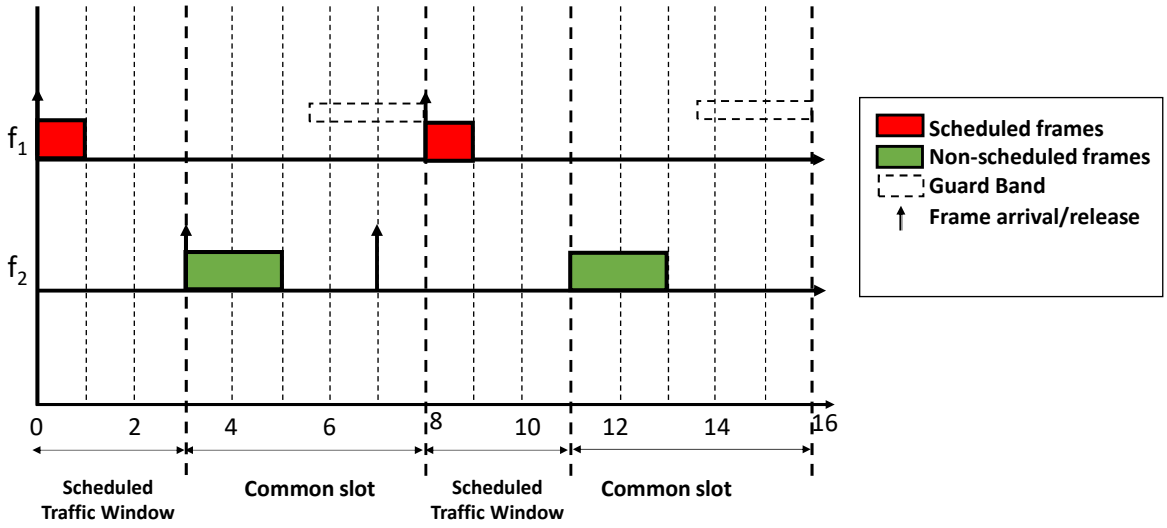


Figure 2.8: Illustration of frame transmission using TAS.

always transmitted in a "scheduled traffic window" and those of f_2 in a "common slot". As can be noted, the second frame of f_2 (arriving at time $t = 7$) cannot start its transmission immediately, although it arrives in a "common slot". The reason for this is the guard band that protects the transmission of the second frame of f_1 from any disruption. We recall that the transmission of a frame cannot begin during the guard band.

Among the drawbacks of this mechanism, it should be noted that while the guard band ensures that the transmission of time-sensitive traffic is protected from blockings, it results in poor use of network bandwidth [2]. To ensure fully deterministic transmission of time-sensitive traffic, the time intervals for each flow along the path must be aligned. This alignment can only be achieved if all nodes have the same notion of time, i.e., if their clocks are synchronized. This synchronization mechanism was described in Section 2.4.1. In addition, the GCLs must be provided, the synthesis of which is also a very computationally intensive task.

> The precomputed GCL is a very rigid approach that is unsuitable for dynamic changes in the network and traffic.

2.4.4.2 Credit-Based Shaper (CBS)

To provide bounded end-to-end delays, moderate the transmission bandwidth of AVB traffic classes, and protect the entire network from burst effects, the AVB Task Group

[2] This is because a frame whose transmission time is greater than the guard band cannot begin its transmission during this period. We note that smaller frames can still be transmitted during the guard band.

introduces the *Credit-Based Shaper (CBS)* – specified in IEEE 802.1Qav (IEEE, 2010). The purpose of this transmission scheme is to guarantee each of the so-called "Stream Reservation (SR)" classes an appropriate share of the link bandwidth, as mentioned in Section 2.3.6. Unlike Strict Priority, CBS prevents the flows of high-priority classes from starving out those of lower-priority classes. For this purpose, a parameter *credit* is defined for each class to regulate the frame transmission for each SR class at the output queues. The selection for head-of-queue frame transmission of an SR class is conditioned by the following: (1) the value of the credit for the class is at least zero; and (2) there are no other higher priority frames available for transmission (Zhao et al., 2018a). The variation of the credit is affected by (*i*) the configuration parameter *Sending Slope* (*sendSlope*), which specifies the rate at which the credit is consumed; and (*ii*) the parameter *Idle Slope* (*idleSlope*), which specifies the rate at which the credit is replenished. As defined in IEEE 802.1Qav (IEEE, 2010), the idleSlope is a user-defined parameter that represents a fraction of the link speed, and the sendSlope is calculated as the difference between the link speed and the idleSlope. The credit of an SR class is reduced by the *sendSlope* when a frame of the class is transmitted. This credit can be replenished in two scenarios: (*i*) when a frame from the class is waiting to be transmitted and another interfering frame from another class is transmitted; and (*ii*) when no frame from the class is in the queue. Non-real-time flows (i.e., best-effort flows) are transmitted when there is no waiting traffic from these classes or when the credits of both classes are negative. As already mentioned in Section 2.3.6, CBS can only guarantee a latency limit of 2 ms or less over 7 hops. We recall that some emerging automotive applications require as low as 500μs – 1ms over several hops (IEEE, 2021). Figure 2.9 shows an example of the CBS transmission process.

In this figure, three flows (f_1, f_2, and f_3) are considered for transmission. We assume that f_1 and f_2 belong to Classes A, and B, respectively, and f_3 is a best efforts flow. The first frames of f_1 and f_2 arrive shortly after the first frame of f_3 start its transmission. Then the class credits of f_1 and f_2 which were initialized to 0, start to increase according to their idleSlopes. As soon as the first frame f_3 completes its transmission, the first frame of f_1 starts its transmission and its class credit starts decreasing according to its sendSlope. Meanwhile, the class credit for f_2 continues to increase until the complete transmission of f_1, when its first frame begins transmission. During the transmission of the first frame of f_2, the class credit for f_1 starts increasing again until its second frame starts its transmission.

An extension of CBS is the *burst limiting shaper* (BLS) (Thiele and Ernst, 2016b). This shaper functions similarly to the CBS, but allows high-priority traffic to make a limited number of excess transmissions in the event of temporary congestion (burst).

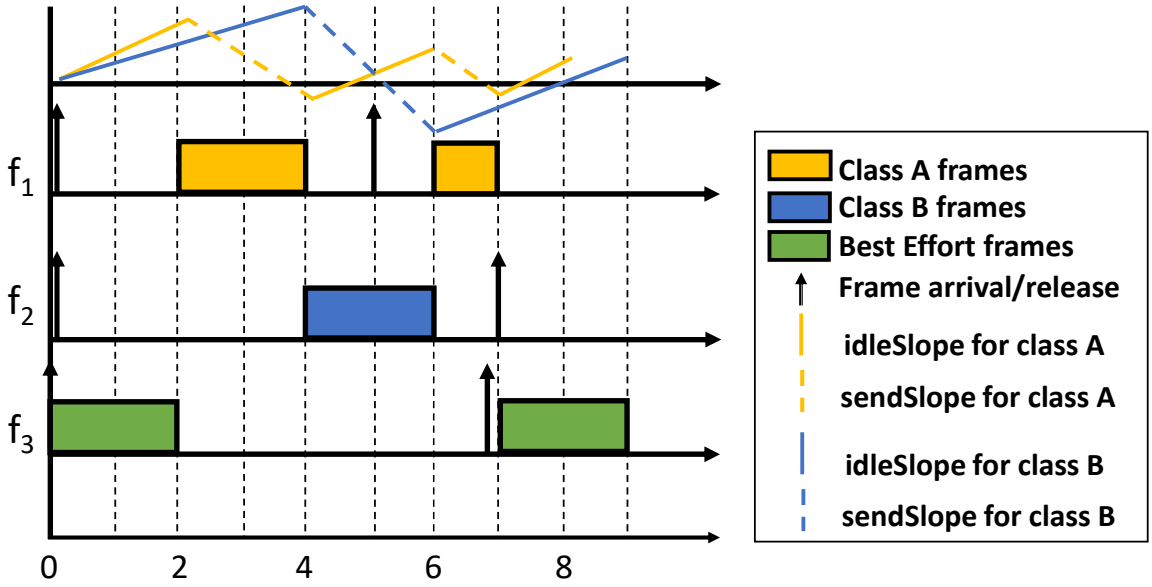


Figure 2.9: Illustration of frame transmission using CBS.

2.4.4.3 Cyclic Queue Forwarding (CQF)

The IEEE introduced the *Cyclic Queue Forwarding (CQF)* (IEEE, 2017a), which organizes the reception and transmission of frames at each node by following well-defined cyclic time intervals. Each time slot is divided into *even* and *odd* intervals. All frames received within one interval are not transmitted until the next interval. Here, any time-sensitive frame received in an even or odd cycle is scheduled for transmission at the next switch in the following cycle. The maximum delay for a time-sensitive frame cannot exceed two cycle times per hop. This makes the network very predictable, since the delay of a frame depends solely on the cycle time and the number of hops. Figure 2.10 illustrates the frame transmission process with CQF.

In this figure, a time-sensitive frame f_1 arrives after a best-effort frame f_2 in the odd interval, but f_1 has been prioritized for transmission in the subsequent even interval. To ensure that the CQF behaves as desired, it is important that the scheduled cycle times for all frames are respected. This means that all transmitted frames must be received by the downstream switch during the expected cycle.

Under CQF, careful consideration must be given to cycle times, the alignment of cycle times between switches in the network, and timing of first and last transmissions within a cycle must be carefully considered to meet timing requirements.

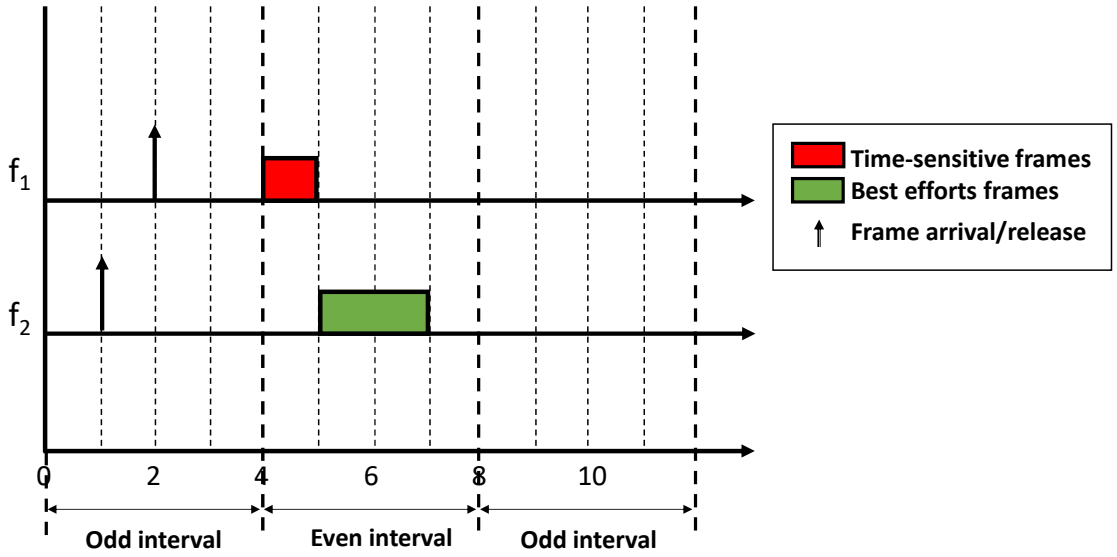


Figure 2.10: Illustration of frame transmission using CQF

2.4.4.4 Asynchronous Traffic Shaper (ATS)

In addition to the TAS, CBS, and CQF transmission schemes, TSN also introduces *Asynchronous Traffic Shaping* (ATS) (Specht and Samii, 2016; IEEE, 2019b) as a traffic control feature. ATS is intended to circumvent the need for a synchronized clock across the network, as is the case with TAS. It uses a queuing and buffering approach to control the flows. It achieves deterministic data transmission in two steps: (1) carefully queuing incoming frames to achieve frame isolation and prevent so-called "*head of line blocking*"; and (2) dequeueing frames for transmission at a controlled rate to ensure that each flow has a fair chance of being transmitted without being discarded or delayed.

> ATS is based on an event-driven model, and transmission decisions are made at each node.

2.4.4.5 Frame Preemption

Frame preemption was defined and introduced in the IEEE 802.3br (IEEE, 2016b) and IEEE 802.1Qbu standards (IEEE, 2016a). These specify a 1-level preemption scheme for Ethernet frames. We have already described the mechanism of this transmission scheme in detail in Section 1.2. Here, we briefly summarize this discussion.

Briefly, two Medium Access Control (MAC) sublayer service interfaces are defined in the link layer: a "preemptable MAC (pMAC) interface" and an "express MAC (eMAC) interface". Frames assigned to the eMAC and pMAC service interfaces are referred to

as "*express*" and "*preemptable*" frames, respectively. Each eMAC frame has a higher priority than any pMAC frame and can therefore preempt any of these frames to speed up its own transmission. Finally, frames of the same preemption class cannot preempt each other. The standards (IEEE, 2016a,b) describe not only the preemption operations, but also the hardware changes required at the switches to support the preemption, which occurs at the network MAC merge sublayer between the physical and MAC layers (see Figure 2.11).

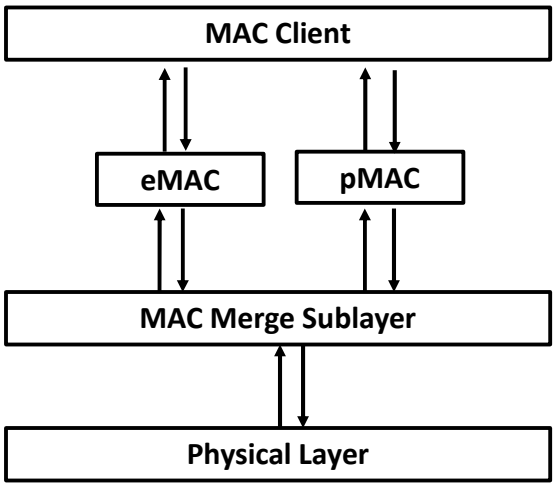


Figure 2.11: The MAC merge sublayer managing service interfaces.

Frames on this sublayer are called Mframes (IEEE, 2016a). Before each Mframe transmission, the sublayer checks whether the next switch/node supports preemption by performing a verification operation described in the standards (IEEE, 2016b). Preemption is not enabled in the sender node until the verification operation confirms that it is supported by the receiver node. If it is, additional information describing the preemption properties is added to the Mframe headers. The sublayer is able to interrupt an interruptible Mframe that is being transmitted and can also prevent a new Mframe from starting transmission (IEEE, 2016b).

Among the advantages of frame preemption over the previously mentioned flow control mechanisms is that it allows more efficient use of available bandwidth, since the transmission of preemptable frames cannot compromise the timing requirements of express frames, and this is guaranteed without the use of a guard band.

Frame Preemption is easy to configure and is particularly suitable for applications where time synchronization is not required (nor applicable), especially for use-cases that require dynamic behavior and/or unsynchronized endpoints, as is common with DRES.

2.5 A closer look at TSN frame preemption

In this section, we discuss the frame preemption feature in detail from four perspectives: (1) the definition of the frame format that enables the preemption operations; (2) the preemptive transmit process as defined in the standards; (3) the preemptive receive process as defined in the standards; and finally (4) the preemption overhead.

2.5.1 Frame format specification

It is important to preserve the Ethernet frame format when Mframes are preempted. To this end, the IEEE 802.3br standard defines Mframe formats in a preemption-enabled environment (see Figure 2.12).

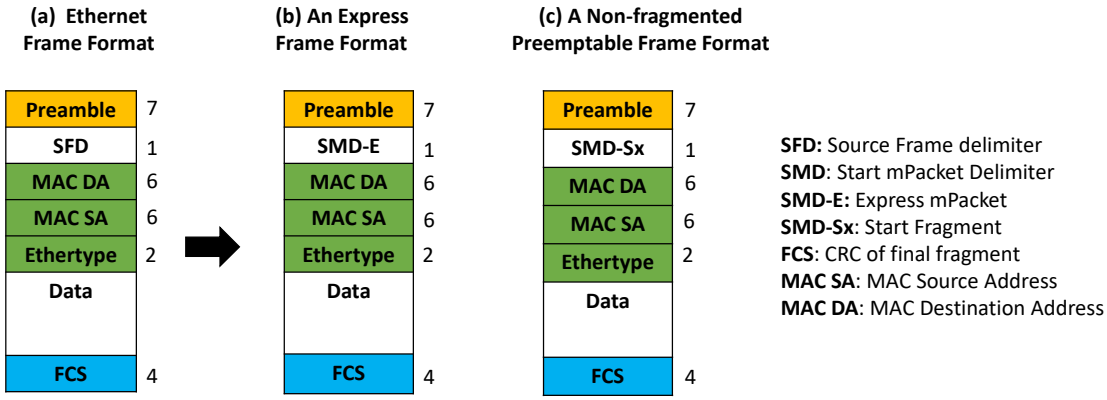


Figure 2.12: Frame formats as specified in IEEE 802.3 Standards where the numbers are in bytes and represent the size of each field.

As can be seen from this figure, the structure of an express frame (see Figure 2.12b) differs from that of the corresponding standard Ethernet frame (see Figure 2.12a) only by one octet, called a "*Start Frame Delimiter*" (SFD). This octet is replaced in the frame format by the "*Start Mframe Delimiter-Express*" (SMD-E). In practice, however, SFD and SMD-E have the same value. Similarly, a preemptable frame that has not been preempted

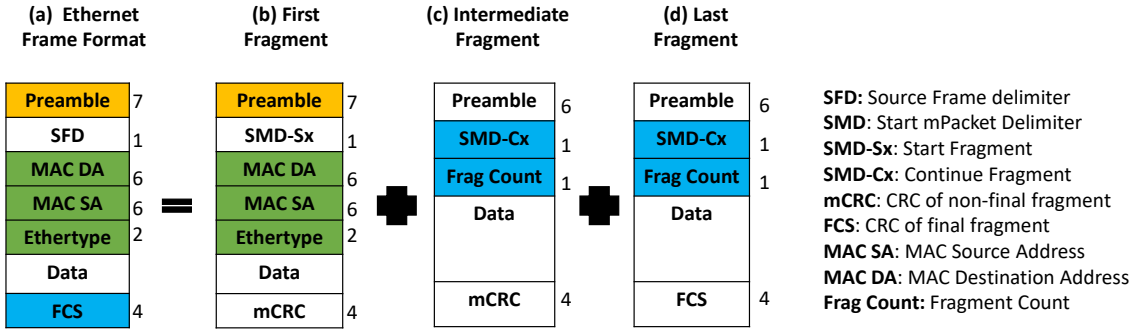


Figure 2.13: Frame fragments formats as specified in IEEE 802.3br Standard where the numbers are in bytes and represent the size of each field.

(see Figure 2.12c) differs from a standard Ethernet frame only by a single octet, here the SFD is replaced by the "*Start Mframe Delimiter Start Fragment*" (SMD-Sx). Now, assuming a frame that has been preempted, it is worth noticing that the first fragment remains almost the same as the original frame as shown in Figure 2.13.

There are only two differences: (1) the *size*, which is smaller (because the original frame has been split into fragments); and (2) the *error checking code* (FCS), which is replaced by a newly generated Mframe error checking code (MCRC) (see Figure 2.13b). All other fragment(s) consist of a frame header containing three components: (*i*) a preamble; (*ii*) a "Start Mframe Delimiter for Continuation fragment" (SMD-Cx); and (*iii*) a so-called "Frag Count", which is used to monitor the correct order of incoming fragments and to detect missing fragments (see Figure 2.13c). All fragments are appended with the same MCRC, except for the last fragment, which ends with the FCS of the original preempted frame (see Figure 2.13d).

2.5.2 Preemptive frame transmission

The *Transmit Processing* function is responsible for frame transmission at the MAC sublayer interface. The process begins with a verification procedure that checks whether the receiver node supports preemption. To do this, it sends a "request frame" to the receiver node to inquire whether it supports preemption. In return, the receiver node sends a response to confirm that it does or does not support preemption. This information is interpreted by the forwarding node based on the SMD value of the response frame. Transmission of a frame with preemption begins only after it is confirmed that both the sender and receiver nodes support preemption. Figure 2.14 shows the detailed *Transmit Processing* state diagram as defined in the IEEE 802.3br Standard. All labels, functions, and variables

are as defined in IEEE (2016b). The transmit process is triggered when a frame reaches the INIT_TX_PROC state.

2.5.2.1 On the transmission of express frames

When an express frame reaches the Transmit Processing function, i.e., at IDLE_TX_ PROC, it is identified as "express" and then transits to the EXPRESS_TX state, which is responsible for non-preemptive transmission. Upon completion, the function transitions to the E_TX_COMPLETE state, where it either returns to the idle state (IDLE_TX_PROC) or resumes transmission of a pending preempted frame.

2.5.2.2 On the transmission of preemptable frames

In contrast to the express frames transmission process, when a preemptable frame reaches the IDLE_TX_PROC state, the Transmit Processing function first checks whether the receiver node has preemption capabilities, and then returns to the IDLE_TX_ PROC state (see connector "C"). The function transitions to the START_PREAMBLE state if the response is positive, which triggers the transmission in a preemptable manner. Note that the transmission can only be interrupted if an express frame arrives and the preemptable transmission has reached a feasible interrupt point. If a preemption occurs, an MCRC value is computed for the preemptable frame fragment (TX_MCRC) and the function transits to a waiting state (RESUME_WAIT). All pending express frames are then transmitted (see connector "B") and transmission of the preemptable resumes only after that. After the transfer is complete, or if preemption did not occur during the frame transmission, the function returns to the IDLE_TX_PROC state. For more detailed information about each state, see pages 48 to 52 of the IEEE 802.3br standard (see (IEEE, 2016b)).

2.5.3 Preemptive frame reception

At the receiver node, a Medium Independent Interface (xMII) checks the SMD for each frame upon arrival, and the value of the SMD indicates whether the frame is an express or a preemptable frame (IEEE, 2014). Express frames (i.e., frames containing SMD-E) are processed by an Express Filter, while preemptable frames are processed by a "Receive processing" construct. This specific Receive processing is responsible for ensuring that all fragments of a preempted frame are received completely and in the correct order. For this purpose, both the "MCRC" and the "Frag Count" values of each fragment are used. The frame transmission of any preempted frame is guaranteed to be complete because all fragments have the same MCRC value, but the last fragment embeds the FCS of the

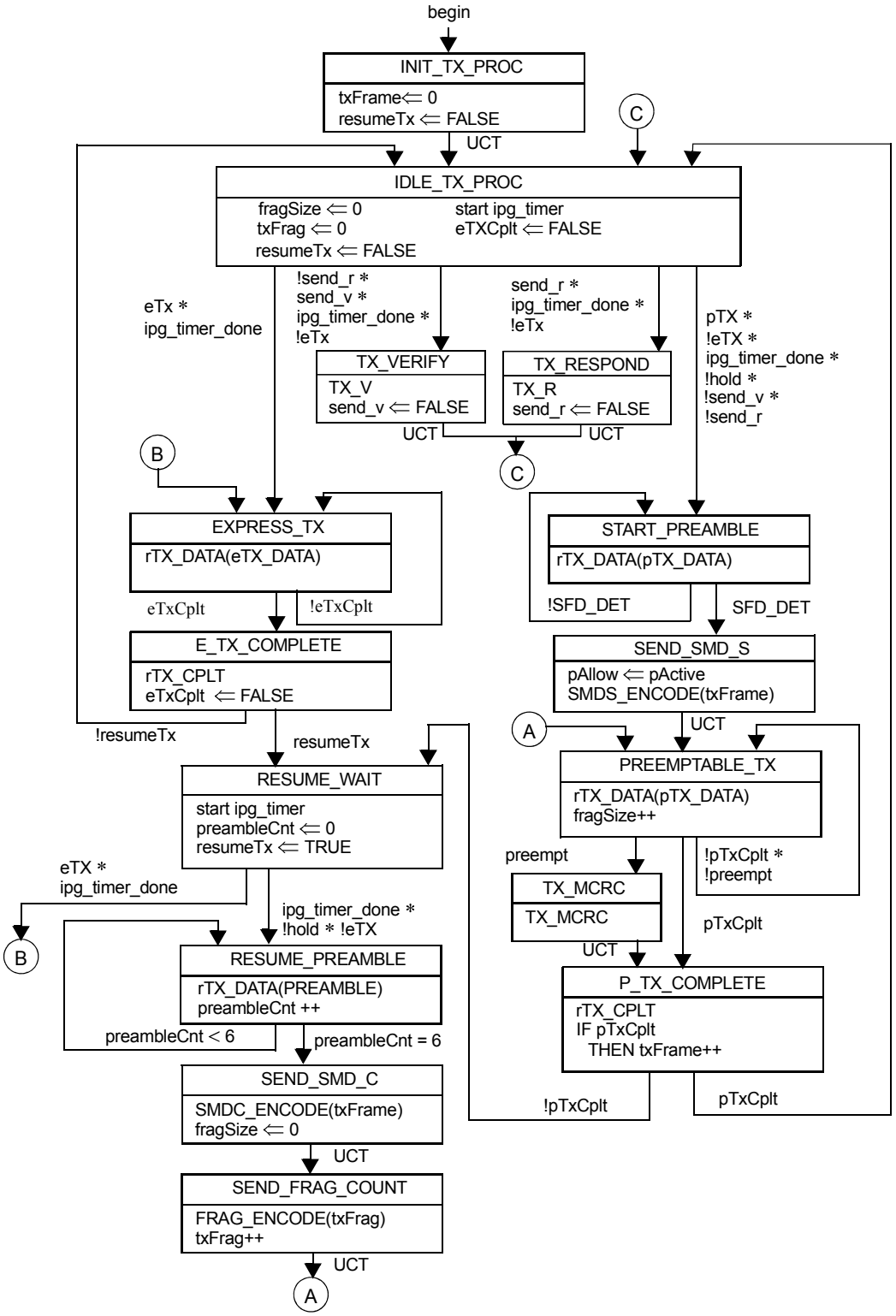


Figure 2.14: Preemptive frame transmission process as specified in IEEE 802.3br Standard.

original frame. This means that the reception of a sequence of fragments belonging to a preempted frame is complete as soon as two consecutive fragments have a different error checking code at the receiver node. Figure 2.15 illustrates the *Receive Processing* state diagram as defined in the IEEE 802.3br Standard (see (IEEE, 2016b), pg. 51). Again, all labels, functions, and variables are defined as in (IEEE, 2016b) (see pgs. 45-48) and the reception process is triggered when a frame reaches CHECK_FOR_START state.

2.5.3.1 On the reception of express frames

When an express frame reaches the Receive Processing function, i.e., at the CHECK_FO R_START state, it is identified as "express" and then transits to the EXPRESS state, which is responsible for receiving it in a non-preemptive manner. Upon completion, the function transits to the IDLE_RX_PROC state.

2.5.3.2 On the reception of preemptable frames

Unlike the express frame reception process, the Receive Processing function enters the CHECK_FOR_START state when a preemptable frame reaches the REPLACE_SFD state, in which the frame's SFD is replaced with a recomputed SMD value. After that, the function transitions to the P_RECEIVE_DATA state, which triggers reception in a preemptable manner. Note that this reception can only be interrupted if an express frame arrives and the preemptable transmission has reached a feasible preemption point. If a preemption occurs, reception is suspended and the function enters the P_WAIT_FOR_DV _FALSE state where it receives the preempting express frame(s). After these frames are fully received, the function transitions to the P_WAIT_FOR _RESUME state, where it resumes receiving the preempted frame. Upon completion or if there was no preemption during frame reception, the function transitions back to the IDLE_RX_PROC.

2.5.4 Preemption overhead

The standards, in their current specifications, do not allow preemption to add any form of *padding* to a fragment of a preempted frame. That is, the data portion of a fragment may not be augmented to meet the minimum Ethernet frame size requirement, which is 84 bytes. To ensure this, the standards command that a preemptable frame must not be preempted until the following two conditions are met: (1) the size of the fragment being transmitted is at least 84 bytes; and (2) the remaining fragment resulting from the occurrence of a preemption operation also meets the minimum frame size. Considering these two requirements, the longest non-preemptable Ethernet frame fragment is 143 bytes long

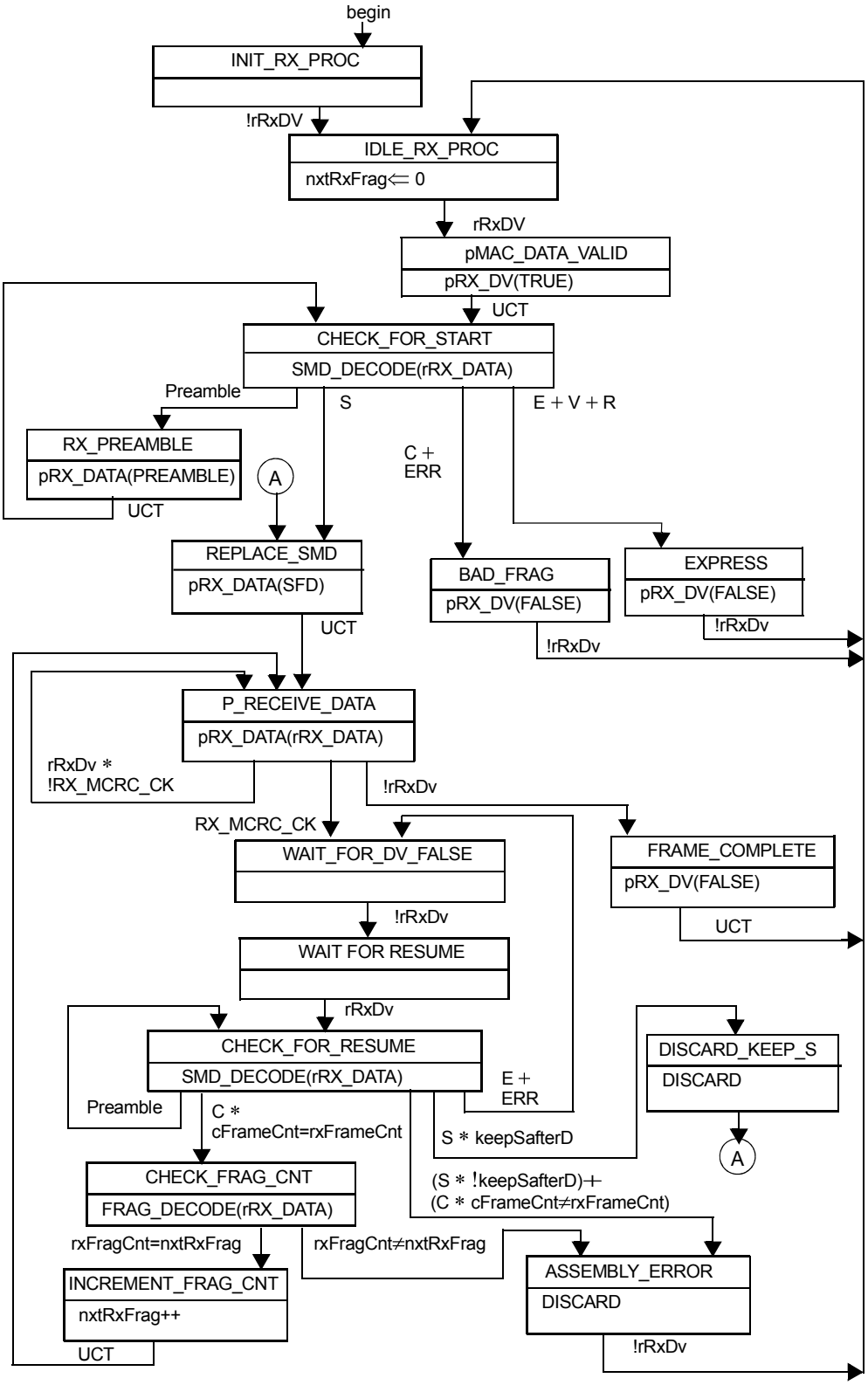


Figure 2.15: Preemptive frame reception process as specified in IEEE 802.3 Standards.

(see Thiele and Ernst (2016a), Lemma 1 for a detailed proof). This means that each express frame can be blocked for at most as long as it takes to transmit 143 bytes of data on the link. This corresponds to blocking of $1.14\mu s$ and $11.44\mu s$, assuming an Ethernet with $1Gb$ and $100Mb$ speed, respectively. Recall that without preemption, the blockings were $12\mu s$ and $120\mu s$, assuming a $1Gb$ and $100Mb$ speed Ethernet, respectively. Roughly speaking, this translates to a reduction of 90.5%, when preemption is enabled. On the other hand, the total overhead incurred by the occurrence of each preemption is 12 bytes (i.e., 6- byte preamble, 1- byte SMD-Cx, 1- byte Frag Count; and finally 4- byte MCRC). In addition, the *Inter Frame Gap* (IFG) between two consecutive transmissions must be accounted for before the next frame/fragment is transmitted. According to the standards, the size of each IFG is equal to the amount needed to transmit 12 bytes of data. Thus, the total overhead associated with the occurrence of each preemption is 24 bytes (i.e., $0.19\mu s$ and $1.9\mu s$, assuming Ethernet speeds of $1Gb$ and $100Mb$, respectively).

2.5.5 Limitations of frame preemption

As discussed in Section 1.2, a very noticeable limitation in the standards is the lack of configuration definitions that guarantee optimal/efficient performance in terms of frame responsiveness. Eight CoS classes are defined in the specification of Ethernet frames, but it is not clear which of them should be mapped to eMAC or pMAC frames in the TSN specification. It also follows from the discussion conducted in Section 1.1 on the 1-level preemption scheme that this scheme significantly improves the responsiveness of eMAC frames. Thiele and Ernst (2016a) has experimentally shown that the 1-level frame preemption scheme improves the performance of express frames. However, the performance of these frames is negatively affected when their number increases (Lo Bello and Steiner, 2019). This tendency is due to the fact that frames of the same class are transmitted in a non-preemptive manner. As a result, frames may have stringent timing requirements but cannot be classified as express to preserve performance. Indeed, real-time traffic in embedded systems may have different timing requirements that do not fit into just one (express) preemption class (Gogolev and Bauer, 2020). In order not to jeopardize the schedulability of these frames, and subsequently the schedulability of the whole system, these frames should not be blocked for excessively long periods of time. The 1-level preemption operation in the current version of the standards does not allow preemptable frames with stringent timing requirements to take advantage of the basic benefits of preemption.

In this dissertation, we proffer solutions to these shortcomings of the 1-level preemption scheme by exploring and promoting the implementation of a *multi-level preemption*

scheme. With this new paradigm, all frames with stringent or firm timing requirements will get a significant improvement in their performance in terms of responsiveness. This, in turn, allows us to improve the QoS of the entire system and increase the workload that can be accommodated in a single communication network. On the other side of the realization of a multi-level preemption scheme, however, are a number of very difficult scientific problems. First, the feasibility of a multi-level preemption viz-a-viz the specification of the standards needs to be investigated. If it is found that a multi-level preemption scheme is feasible, a framework to model, analyze, optimize, and predict the temporal characteristics of such a scheme must be developed. Finally, the performance gains that the scheme brings must be measured and quantified. Our contributions, which we will present in the following chapters of this dissertation, address the above challenges. Before proceeding with the presentation of these contributions, we present related work in TSN frame preemption research in the next section.

2.6 Related Work

Several works have studied the TSN frame preemption feature from different perspectives. In the following subsections, we have grouped the discussion of related work with respect to the challenges that we address in this thesis.

2.6.1 Feasibility and implementation cost comparison

Kim et al. (2013) showed that preemptive switched Ethernet offers better performance than the standard IEEE 802.1Q/p protocol in terms of end-to-end transmission delays. This is especially true when real-time and non-real-time frames are transmitted over the same network. Thiele and Ernst (2016a) also showed, still by simulation, that the end-to-end delays of express frames under preemptive Ethernet are very close to those of TAS, suggesting standard preemptive Ethernet as a promising alternative to TAS in scenarios where long CAM frames are not making use of the express class.

As for the hardware implementation, Zhou et al. (2017) presented a *VHSIC Hardware Description Language* (VHDL) design layout for the transmit and receive processes, as well as an FPGA-based hardware implementation of TSN sender and receiver nodes. By using a *Field-Programmable Gate Array* (FPGA) based implementation, Hotta et al. (2015) also provided a quantitative evaluation of the performance improvement associated with the preemption phase was also performed. The measurement results show that the maximum latency of express frames was significantly reduced (from 27.57μs to 2.46μs @1Gb/s). Other authors such as Jia et al. (2013); Simon et al. (2017) have also pointed

out some challenges that could arise from preemption operations, such as starvation of low priority frames and buffer overflow (when there are more fragments of preempted frames to store than the buffer size of the switches/nodes). However, these authors have neither highlighted nor solved the problem of multi-level preemption.

There is a plethora of research comparing TSN traffic control approaches from a performance perspective. In this context, Thangamuthu et al. (2015) compares the performance of three TSN shapers, namely TAS, Burst Limiting Shaper and the Peristaltic Shaper (PS). In their work, the comparison metrics are delay and jitter. The authors conclude that TAS offers the best performance on both metrics. A similar work by Thiele et al. (2015a) compares TAS with PS and comes to a similar conclusion, i.e., TAS provides better end-to-end delay performance. However, they found that the performance of TAS deteriorates significantly when the end systems are not synchronized.

Another work comparing TAS and frame preemption, by Gogolev and Bauer (2020), has shown through experiments that frame preemption is better for industrial networks with unsynchronized end systems. The authors point out that a serious limitation for frame preemption is the low Quality of Service (QoS) resolution, i.e., there are only two classes (eMAC and pMAC) specified for preemption in TSN. On another front, a work by Nasrallah et al. (2019b) does a performance comparison between TAS and Asynchronous Traffic Shaping (ATS). The authors conclude that ATS compares favorably with TAS for sporadic (asynchronous) traffic. However, they find that the performance of Scheduled Traffic degrades under ATS as the best-efforts load increases. We note that none of the above work has compared these features from an implementation cost perspective.

However, there is not such a strong interest in comparing the performance of CBS with other shapers. There are works by Alderisi et al. (2013) and Bello (2014) that present simulation results for AVB-ST (Scheduled Traffic) in industrial automation and automotive applications, respectively. AVB-ST is a more rigorous approach than TAS, assuming that the scheduled traffic is strictly periodic and that precise offsets are provided for the scheduled frames. In these works, the baseline is considered to be pure CBS and the authors note that only by using AVB-ST is it possible to guarantee bounded delay and zero jitter for the scheduled traffic. A similar work by Meyer et al. (2013) highlights the importance of the quality of schedule tables. For example, with a non-dense table, the delay of AVB traffic increases but is still bounded and complies with the specification in IEEE (2016e). When a compact schedule table is used, the delays of AVB traffic are still bounded, but no longer conform to the specification.

2.6.2 Model and analysis

Most of the work in the literature on frame preemption in Ethernet have focused on the impact of preemption on end-to-end frame transmission delay. Simulations are usually used for this purpose (see, for example Jia et al. (2013); Kim et al. (2013); Thiele and Ernst (2016a); Zhou et al. (2017) for more details). Most authors relying on this methodology acknowledge the effectiveness of frame preemption in Ethernet and have concluded that it allows system designers to drastically reduce the transmission delays of express frames while not significantly degrading the performance of preemptable frames. However, it is well known in the research community that simulation is neither a comprehensive nor a rigorous means of evaluating the performance of a system. This is because it does not guarantee that the case that causes the worst-case scenario will occur. Consequently, despite the extensive quantitative performance results obtained with this technique, more sophisticated approaches are needed to provide guarantees on the end-to-end delays of Ethernet frames. Network Calculus (NC) (Reimann et al., 2013); Trajectory Approach (TA) (Martin and Minet, 2006a); and Compositional Performance Analysis (CPA) (Henia et al., 2005) have all been used as established techniques to provide timing guarantees for real-time Ethernet flows.

In NC, the so-called *arrival curve* and *service curve* are used to model the arrival of flows and the transmission bandwidth at a switch output port, respectively. To our knowledge, there are only a handful of papers in the TSN-related literature that use this approach. Zhao et al. (2018b) provided a worst-case latency analysis for IEEE 802.1Qbv networks. For this purpose, they assumed that the Gate Control List (GCL) and priority assignment configurations are given. They validated the performance of their approach against synthetic and real-world use cases in terms of scalability and the impact of GCL overlap features on individual flows. In another work, the authors also performed latency analysis for AVB traffic in TSN with NC (Zhao et al., 2021). However, their analyzes are only for non-preemptive TSN networks and leave the preemptive case unanswered.

In TA Martin and Minet (2006a) examined the highest number of frames sharing the same trajectory, as this is a potential source of delay for each of these flows. The approach adopted proceeds “backwards”, i.e., from the receiver node to the source node. In another context, this approach was used by Bauer et al. (2012) for timing analysis of AFDX with strict priority, non-preemptive flows transmitted according to a FIFO scheduling strategy. In the work, TA was further improved by investigating the basic idea that flows using a common link cannot arrive at a switch at the same time. Li et al. (2014) have proved the result to be optimistic and corrected the flaw. Nevertheless, the analysis still considers only non-preemptive frame transmission.

Cao et al. (2016a,b) introduced a so-called *eligibility interval* approach for timing analysis of Ethernet Audio Video Bridges (AVB) (IEEE, 2011a) Networks with Credit-Based Shapers (CBS). This approach examines the worst-case performance of a flow when a flow has some pending payload to transmit and has a non-negative transmission credit. This approach has been shown to be tight for AVB networks and has subsequently been extended for timing guarantees of AVB flows in standard TSN (Maxim and Song, 2017). Thangamuthu et al. Thangamuthu et al. (2015) proposed a worst-case performance analysis for Switched Ethernet with Burst-Limiting Shaper (BLS); Peristaltic Shaper (PS); and Time-Aware Shaper (TAS) for TSN, but concluded that only the TAS can schedule control traffic within the maximum delay imposed by the Standards.

CPA has been used extensively for the timing behavior of Ethernet flows (Thiele et al., 2015b, 2014). It uses the so-called *level-i busy period* approach, activated by the so-called critical instant, to study the worst-case response time of each real-time flow (Hofmann et al., 2017). CPA was used for the timing behavior of Switched Ethernet in (Rox and Ernst, 2010) and for Ethernet AVB in (Diemer et al., 2012a). The analysis for Switched Ethernet was improved by Thiele et al. (2014) to tighten the worst-case response time bounds of each flow by up to 80%, then the same authors exploited the FIFO nature of Switched Ethernet transmission (Thiele et al., 2015b) to reduce interference estimates in frame transmissions and achieved a latency improvement of about 30% over the then-current state-of-the-art in CPA. Nevertheless, Thiele et al. (2015a) and Thiele and Ernst (2016b) proposed worst-case analysis for TSN using CPA with PS and BLS, respectively. Both contributions focused only on shapers, leaving frame preemption concerns beyond the scope of their work. On another front, Thiele and Ernst (2016a) used CPA to provide worst-case guarantees for both Standard Ethernet and IEEE 802.1Qbv when frame preemption is enabled. Lo Bello et al. also provide a schedulability analysis for IEEE 802.1Qbv networks with preemption support (Lo Bello et al., 2020a). In these works, the authors only address the traditional 1-level preemption scheme as defined in the standards.

Recently, Knezic et al. (2020) investigated the multi-level preemption from an implementation standpoint. However, their work falls short of providing a formal analysis of the worst-case performance guarantees. In this work, we fill this gap by providing a worst-case analysis of the traversal time of TSN frames under the multi-level preemption scheme, which to our knowledge is the first contribution in this direction.

2.6.3 Network configuration

An appropriate priority assignment policy plays a central role in the resulting performance of real-time systems (Davis et al., 2016). For example, in CAN, the maximum

reliable utilization was initially thought to be 35% because message IDs (corresponding to message priorities) were assigned randomly or ad hoc (Buttle, 2012). Davis et al. (2016) later showed that Audley's Optimal Priority Assignment (AOPA) algorithm Audsley (2001) can achieve reliable CAN utilization of over 80%. AOPA has become the reference scheme for priority assignment in many real-time systems and has been shown to be optimal when there are no priority inversions (Park and Shin, 2019). Davis and Burns (2009) has introduced an improvement to AOPA – the Robust Priority Assignment algorithm (RPA) – which, in addition to being optimal, also maximizes the number of tolerable transmission errors in CAN. We note that both AOPA and RPA are not applicable to IEEE802.1 Qbu networks, as priority inversion may occur in these networks (Ojewale et al., 2020).

For systems where priority inversions may occur, Deadline Monotonic Priority Ordering (DMPO) and Deadline minus Execution time Monotonic Priority Ordering (D-CMPO) and the so-called "DkC" heuristic are commonly used, and it is well known that these heuristics dominate most other priority assignment heuristics in the literature in terms of schedulability (Davis et al., 2016; Lee et al., 2021). In particular, DMPO is the recommended priority assignment scheme in scenarios where preemption overheads are considered and/or fixed-priority scheduling with preemption thresholds. These two conditions also apply to IEEE802.1 Qbu networks, since each preemption introduces significant overhead, and thresholds are set for each preemption class (Ojewale et al., 2020). On the other hand, DMPO is not suitable for priority assignment in preemptive TSN because it provides a fully ordered priority list for the flowset, but Ethernet supports only up to eight priority levels. Consequently, this leads to another bin packing problem, which is known to be strongly NP-Complete.

In TSN, most of the work in the literature has focused on the configuration synthesis of time-triggered Ethernet networks, where flows are transmitted according to a pre-computed schedule. In this context, Beji et al. (2014) has proposed a Satisfiability Modulo Theories (SMT) approach and Tamas-Selicean et al. (2012) has proposed a heuristic for the communication synthesis of TTEthernet (Steiner et al., 2009). On another front, Serna Oliver et al. (2018) and recently Reusch et al. (2020) have proposed various frameworks for synthesizing the so-called Gate Control Lists for the IEEE 802.1 Qbv (IEEE, 2016c). For event-triggered time-sensitive Ethernet networks, Specht and Samii (2017) considered a TSN network with the Urgency Based Scheduler (UBS) and used an SMT approach to assign hard real-time flows to queues and priority levels to these queues on a per-hop basis. This work differs from ours in that it addresses the priority assignment for UBS only and assumes a non-preemptive frame transmission scheme. Gavriluţ and Pop (2020) provided a method for assigning traffic classes to frames

in TSN-based mixed criticality systems. However, none of these works addressed priority assignment and configuration synthesis of preemptive TSN networks.

Several works have applied Machine Learning (ML) techniques to various problems in the real-time domain (Ae and Aibara, 1990; Cardeira and Mammeri, 1994, 1995; Lee et al., 2021). Most relevant to this work is the recent work of Lee et al. (2021), in which the authors proposed a Priority Assignment Learning (PAL) framework for multi-core real-time systems. PAL was found to be more effective than existing approaches but suffers from severe scalability challenges as the number of tasks grows. There are other works in the literature that have applied ML techniques to TSN (Mai et al., 2019a,b; Mai and Navet, 2021; Navet et al., 2019). Mai et al. (2019b) and Navet et al. (2019) employed ML techniques to search for feasible TSN configurations. We note that both works do not address the priority assignment problem as part of the configuration. The authors also presented a so-called "hybrid" approach that combines ML techniques with theoretical performance analysis to control the false prediction rate of ML models (Mai et al., 2019a). In addition, Mai et al. recently presented a Generative Neural Network (GNN)-based technique for predicting a feasible TSN configuration (Mai and Navet, 2021). But the work stops short of defining any priority assignment scheme.

Park et al. (2019) showed that both the priority and preemption class assignment schemes used in a preemptive TSN network have a significant impact on the ability of frames to satisfy their timing constraints. In this regard, the authors proposed a framework to compute efficient priority assignments for flows and an efficient eMAC/pMAC queue boundary at each switch port. However, that work assumes a 1-level preemption scheme, leaving open the question of an appropriate priority assignment policy, preemption levels, and flow-to-preemption-class assignment. These gaps are filled in this thesis.

2.6.4 Flow routing

Routing time-sensitive flows is non- trivial (Nasrallah et al., 2019a). Routing optimization has been well studied in the literature and sophisticated techniques have been proposed (Wang and Hou, 2000; Grammatikakis et al., 1998). However, contributions on TSN routing schemes have started less than a decade ago. In this context, both the rapid spanning tree protocol and the shortest path bridging algorithms have been widely adopted in practice (Pop et al., 2016). On the other hand, the IEEE802.1 Qca standard (IEEE, 2016d) specifies the Constrained Shortest Path First routing algorithm for TSN transmissions, but this algorithm does not prevent congestion situations and may increase contention in the network. Arif and Atia (2016) proposed a methodology to evaluate the

routes of a TSN end-to-end connection, but load balancing was not part of their objectives.

Nayak et al. (2018) investigated Integer Linear Programming (ILP) based algorithms for routing time sensitive flows in TSN networks with Time Aware Shapers. The proposed approach in their work differs from ours in that it does not consider the problems of congestion and load-balancing. Aiming at better load balancing for a TSN network, Singh (2017) has presented a meta-heuristics based algorithm capable of routing new traffic flows at runtime with minimal overhead. However, the proposed approach uses the shortest path algorithm (SPA) as the initial solution and does not consider all feasible routes. This limits the solution search space. Gavrilut et al. (2017) also took the same path and proposed heuristic methods for topology and routing synthesis. Their method attempts to achieve optimal utilization of switches and links and efficient routing of flows. However, they did not consider load-balancing. This thesis fills this gap: It addresses the problem of load balancing, disjoint routing for duplicate flows, and dynamic re-routing in congestion situations.

Chapter Summary

In this chapter, we have given a detailed overview of real-time communication and the evolution of communication technologies. We introduced Ethernet and explained how various Ethernet-based technologies have been developed for real-time communications. Then we introduced TSN, the latest set of IEEE standards that add more real-time capabilities to Ethernet technologies. We then focused on a prominent TSN feature - Frame Preemption - and described its specifications in the standards. In particular, we examined the transmit and receive functions as they are currently defined in the standards and highlighted the limitations of the current specifications. Finally, we have looked at related research on frame preemption.

Now that all related concepts and technologies have been introduced, we will present the contributions of this dissertation in the following chapters.

Chapter 3

From 1-level to Multi-level Preemption

In this chapter, we critically examine the feasibility, overheads, and changes required to enable multi-level preemption in TSN. To investigate the possibility of additional preemption levels, we assume a special case where three frame classes and two preemption levels are considered. In Section 3.1, we explore the dynamics of the 1-level preemption scheme as defined in the standards to assess the feasibility of enabling more than one level of preemption and propose modifications and implementation recommendations to enable multi-level preemption. We do this at both the frame transmission level (see Section 3.1.1) and the reception level (see Section 3.1.2). We also discuss interoperability (see Section 3.1.3) and frame buffering (see Section 3.1.4) in the multi-level preemption scheme. Finally, in Section 3.2, we quantify the implementation cost of multi-level preemption in terms of hardware overheads and compare this cost with that of other TSN flow control mechanisms. In particular, we discuss the hardware development and comparison metrics (see Section 3.2.1), the comparison methodology and context (see Section 3.2.2), and the evaluation results (see Section 3.2.3).

3.1 Feasibility of Multi-level Preemption

To achieve multi-level preemption in TSN, it is necessary that the switch nodes are able to identify more than two classes of frames. That is, each switch node in which preemption is enabled should be able to distinguish frames that belong to a different class than the traditional eMAC and pMAC classes. To achieve this goal, we need to extend the definition set of the SMD octet used to determine whether a frame is preemptable or not at the MAC merge sublayer. In this way, it would be possible to further partition the set of preemptable frames. In their current specifications, the standards define 11 different SMD values (IEEE, 2016b). These values not only allow the distinction between eMAC

and pMAC frames, but also describe the verification frames (i.e., the frames that are sent to determine whether the next node supports preemption). In addition, the current specification of preemption in the standards does not allow frames belonging to the same class to preempt each other (IEEE, 2016b). To maintain this convention, and especially for interoperability reasons (discussed later in this document), it is important to define an additional MAC Merge sublayer interface to support each additional level of preemption, as shown in Figure 3.1.

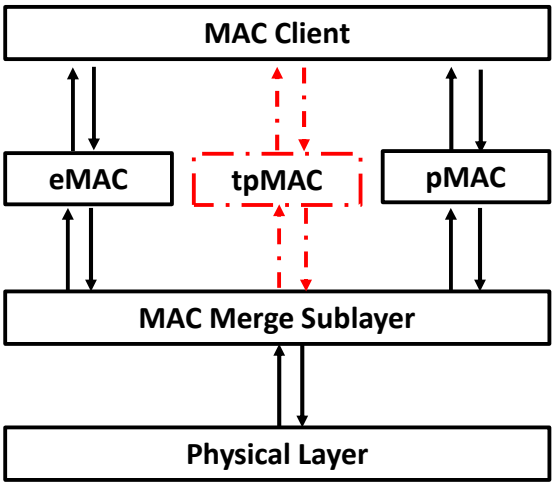


Figure 3.1: Modified MAC merge sublayer with an additional preemptable service interface named *time-sensitive preemptable MAC* (tpMAC). Added features are represented in red color

In this figure, an additional preemptable service interface called *time-sensitive preemptable MAC* (tpMAC) is introduced between the eMAC and pMAC interfaces to handle the so-called *time-sensitive preemptable frames* – a subset of preemptable frames with firm timing constraints. This ensures that frames of any class can be assigned to a unique MAC merge sublayer interface. In addition, both the *Transmit Processing* and *Receive Processing* functions would need to be modified before multi-level preemption could be enabled. We have described these two processes in detail in Section 2.5. Therefore, in the following sections, we will focus only on the proposed changes to support multi-level preemption.

3.1.1 Frame transmission under a multi-level preemption scheme

For frame transmission under a multi-level preemption scheme, we need the SMD values for the verification process. Recall that the current SMD values are defined to inform the

sender whether or not the receiver supports frame preemption. However, in the case of multi-level preemption, the receiver requires new SMD values that represent the number of preemption levels that the receiver node supports. Figure 3.2 illustrates a *Transmit Processing* state diagram to support an additional preemption level that extends the basic 1-level preemption scheme presented in Figure 2.14.

In this figure, all new proposed transitions and/or modules are marked with red dashed lines. Below is a description of how they work. All labels, functions, and variables are as defined in the standard (see pgs. 45–48).

▷ **On the proposed modifications for enabling an additional level of preemption**. Adding an additional level of preemption does not require any change to the transmission process of express frames, since we are concerned with the transmission of preemptable frames with firm timing requirements. New states must be added for such frames. In Figure 3.2, we define two preemptable MAC merge sublayer interfaces, denoted p_1 and p_2 (corresponding to tpMAC and pMAC in Figure 3.1, respectively), and enforce the following rules.

- Any p_1 frame can preempt any p_2 frame, but the converse is not true.
- Upon preemption, any p_2 frame can continue its transmission only if all pending express and p_1 frames/fragments have completed their transmission.

With this new preemption level, we must not only check if the receiver node has preemption capabilities, but also check its preemption level (0, 1 or 2). When a preemptable frame reaches the state IDLE_TX_PROC, then the function (*i*) transitions to the state START_PREAMBLE if it is a p_1 frame; or (*ii*) transition to the newly defined state START_PREAM BLE_2 if it is a p_2 frame.

> In this context, each p_1 frame is transmitted in a similar manner to a pMAC frame in the present specification, while each p_2 frame is transmitted in such a way that it can be preempted by both express and $p_1$frames. If preempted, the p_2 frame is not resumed and sent to completion until all pending express and p_1 frames/fragments have been completed.

3.1.2 Frame reception under a multi-level preemption scheme

Similar to transmission, adding an additional level of preemption does not require any change to the reception process of express frames. This is because we are only concerned

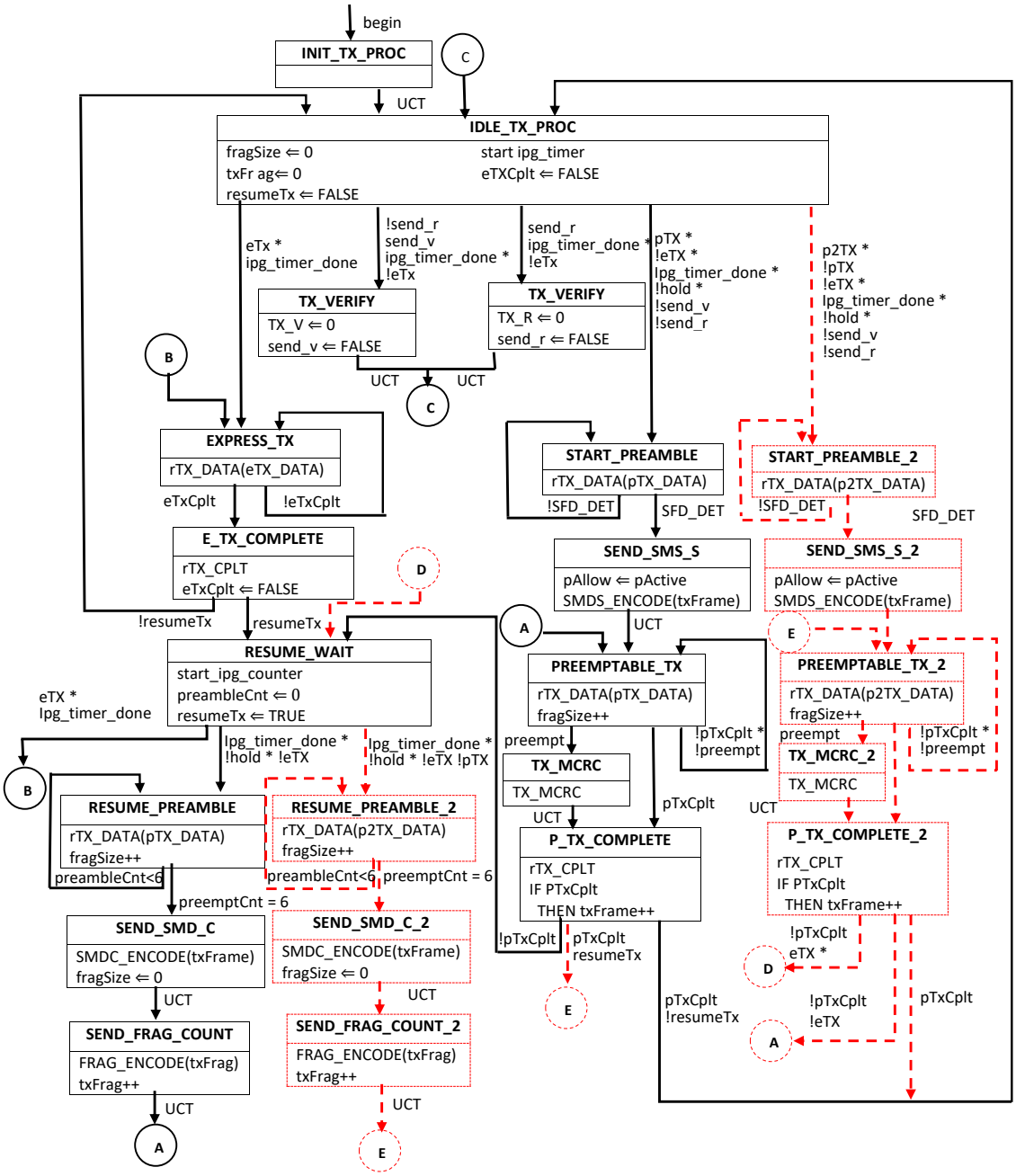


Figure 3.2: Modified Transmit Processing state diagram for two-level preemption. Added features are represented in red color

by the reception of preemptable frames with firm timing requirements. For such a frame, new states must be added. The changes are shown in Figure 3.3 with red dotted lines.

In this figure, we also define two preemptable MAC Merge Sublayer interfaces, denoted p_1 and p_2, and enforce the following rules.

- Any p_1 frame can preempt any p_2 frame, but the converse is not true.
- Upon preemption, any p_2 frame can resume its reception only when the reception of all pending express and p_1 frames/fragments is complete.

With this new preemption level, the receiver node not only confirms that it has preemption capabilities, but also its preemption level (0, 1, or 2). When a preemptable frame reaches the CHECK_FOR_START state, the function (*i*) transitions to the P_RECEIVE _DATA state if it is a p_1 frame; or (*ii*) to the newly defined P2_RECEIVE_DATA state if it is a p_2 frame.

> In this context, each p_1 frame is received in a similar manner to a pMAC frame in the present standard specification, while each p_2 frame is received in such a way that it can be preempted by both express and p_1 frames. When a preemption occurs, reception of the p_2 frame can only continue when reception of all pending express and p_1 frames/fragments is complete.

At this point, we have described the changes required to enable multi-level preemption. In addition to these changes, there are other important operational factors, i.e., *interoperability* and *frame buffering*, that need to be reconsidered before a full roll-out of nodes with multi-level preemption. We address these issues in the following subsections.

3.1.3 Interoperability

In practice, nodes with more than one level of preemption will coexist with other nodes that support only one level of preemption or no preemption at all. This coexistence can be ensured through the verification process in the Transmit Process function mentioned earlier. After a forwarding node sends a verification request to verify that the receiving node supports preemption, the response should include not only that information but also the level of preemption that the node supports. Similar to the 1-level preemption scheme, multi-level preemption is only enabled if the receiver node supports multi-level preemption.

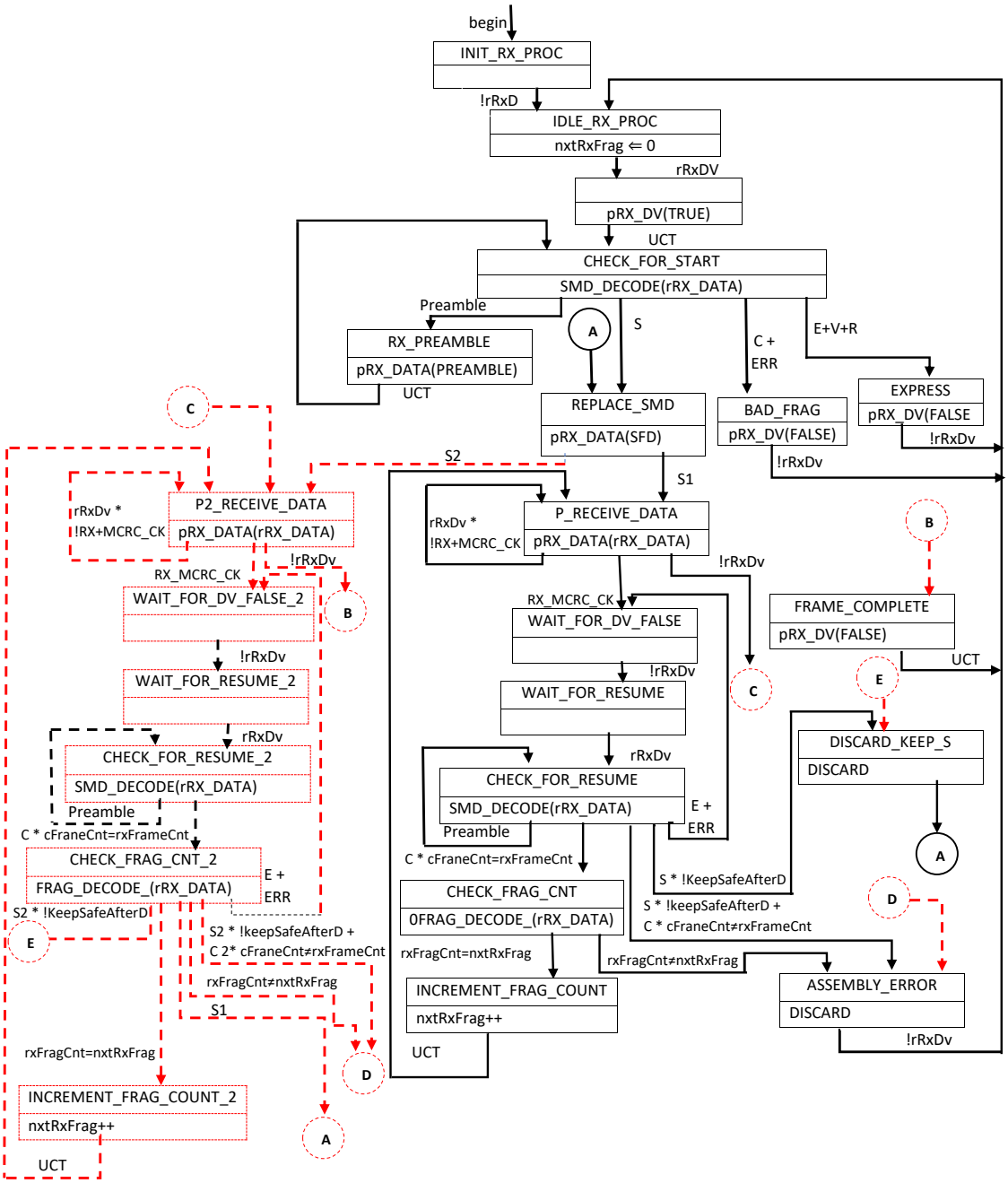


Figure 3.3: Modified Receive Processing state diagram for two-level preemption. Added features are represented in red color

If k-level preemption (with $k \in \mathbb{N}$) is supported, the forwarding node will transmit the frames in a k-level preemptive manner.

3.1.4 Frame buffering

Multi-level preemption requires careful attention to how input buffers are used when receiving incoming frames. We recall that the reception of a sequence of fragments belonging to a preempted frame is completed as soon as two consecutive fragments have a different error-checking code at the receiver node. Once this mismatch occurs, the receiver node assembles the contents of its buffer as a single frame. In the case of multi-level preemption, the mismatch may also result from the reception of a preemptable frame of a higher preemption class. In this case, there are two sets of preemptable frame fragments at the receiver node's input port.

We recommend that frames of different preemption classes be received in different input buffers. This would guarantee the integrity of each frame during preemption operations.

3.2 Implementation cost

As part of the feasibility analysis of multi-level preemption, we quantify the hardware implementation costs of the scheme and compare them to those of two other widely studied TSN traffic control mechanisms, namely TAS and CBS (see their detailed descriptions in Sections 2.4.4.1 and 2.4.4.2). Most studies in the literature that compared these mechanisms did so mainly from a performance perspective (Gogolev and Bauer, 2020; Thiele and Ernst, 2016a; Hellmanns et al., 2020; Nasrallah et al., 2019b) and focused on the worst-case delay and jitter guarantees of frames. However, a detailed discussion of how these mechanisms compare implementation costs were still lacking. System designers are interested in both achieving acceptable performance and reducing hardware costs by choosing simpler and less complex solutions. As long as the performance requirements are met, businesses will opt for the cheapest solution. Comparing implementation costs would help system designers decide which feature(s) to use for their applications. If the cost of implementing multi-level preemption is significantly lower than the other traffic control mechanisms, this is another compelling reason to adopt it.

When estimating switch manufacturing costs, device manufacturers use the term *non-recurring engineering (NRE)* for one-time costs and *recurring engineering (RE)* for costs that are repeated for each device (e.g., testing) (Pruski et al., 2021). In our context, NRE costs are the most appropriate metric because they have the most to do with the hardware components of the switch. To compare NRE costs, a survey of well-known vendors on the development costs of the TSN features would be ideal. However, due to the sensitive nature of this information, vendors are unwilling to disclose it. Instead, we focus on metrics that are easier to obtain, such as the FPGA resource utilization profile from our collaborators at Comcores ApS, which is described in more detail in Section 3.2.1.

> Our comparison is based on the assumption that more resources used means a more costly implementation.

3.2.1 Hardware development and comparison metrics

To compare the cost of a hardware implementation of TAS, CBS, and Frame Preemption, we use resource utilization reports for hardware modules executing the flow control features. These modules are implemented in Xilinx FPGA devices. The reports are obtained during the digital design process (described in Section 3.2.1.1) and include the number of basic hardware blocks (described in Section 3.2.1.2) required for the implementation of each feature.

3.2.1.1 Digital design process

The TSN modules are implemented as Register Transfer Level (RTL) Silicon Intellectual Property (IP) Cores. The source code of these IP cores, written in a Hardware Description Language (HDL), is then processed by Computer-Aided Design (CAD) tools (Brown and Vranesic, 2014) to:

1. express the design with logic gates (*logic synthesis*),
2. map the obtained logic circuit to components specific to a particular implementation technology, and then place and route them on either a silicon wafer or a subtype of Programmable Logic Devices (PLDs) *(physical design)*.

One of the outputs of the *physical design* is the *size* of the implemented module. For Application Specific Integrated Circuits (ASICs), the physical design maps the logic circuitry to cell libraries, and the end result for size is simply an area on a wafer.

In FPGAs, which are a subtype of PLDs, the devices already have a fixed number of resources that the circuit is mapped to. The size is then expressed as the number and type of resources used.

3.2.1.2 Xilinx FPGAs

At a high level, a basic FPGA device consists of Logic Blocks, Programmable Interconnect, and I/O pins (Brown and Vranesic, 2014). In Xilinx devices, the key resources used to implement sequential and combinational circuits are Configurable Logic Blocks (CLBs). These blocks contain Look-Up Tables (LUTs) (an n-input truth table) for combinatorial logic and memory elements for sequential logic (CLB registers). The CLBs also contain dedicated carry logic for arithmetic operations (CARRY8) and Multiplexers (FnMUX) to maximize resource utilization within a block. For more information on FPGA resources as well as modern FPGA architecture, we refer the interested reader to (Xilinx, 2017, 2020). In addition to CLBs, some FPGA devices contain specialized blocks and hard IPs (non-programmable modules that perform a specific function, such as transceivers). These specialized blocks include:

- Block Random Access Memories (BRAM), which contain arrays of Static RAM (SRAM) cells and sometimes FIFO logic,
- Digital Signal Processing (DSP) blocks, which are used to perform more complex arithmetic operations,
- Clock tiles, which contain primitives for clock generation and buffering.

All of our synthesis runs were performed for the Xilinx Zynq UltraScale+ family device `xczu9cg-ffvb1156-1-e`, as this device is used in the popular ZCU102 boards.

3.2.2 Comparison methodology and context

To accurately contextualize the costs added by each TSN feature, we have chosen an absolute baseline to be the whole `xczu9cg-ffvb1156-1-e` device. We note that a basic switching system without TSN features could also be used for this purpose. However, since the size of a switching system can vary significantly across different architectures and configurations, the entire FPGA device is a better reference point. We also compare the TSN features with each other. This comparison is enabled by a weakly coupled

and modular design. In such a design, the queuing and buffer systems are common for TAS and CBS. Therefore, the memories and logic in them can be excluded from the comparison. In contrast, the comparison of Frame Preemption and TSN Shaper presents an additional challenge. The TSN MAC must have some buffer capacity at the receiver nodes to correctly reassemble the frame fragments. The size of these buffers depends on the maximum frame size, but we assume that it is minimal compared to the buffers in the switch output buffers (queues). Therefore, the buffers in the TSN MAC are also excluded from the comparison.

The results obtained have the disadvantage that they only approximate the NRE cost by capturing the final *size* of the product. They do not capture the RE costs associated with testing each chip. This approximation is made possible by the assumption that the larger the device, the longer and more difficult the development. This assumption is at odds with the fact that the optimization phase of development aims to reduce area/resources. When a significant amount of development effort is put into this phase, the result is a *smaller* device. Nevertheless, it can be assumed that the degree of optimization is similar within an organization, so we can use the resource utilization reports for our comparison.

3.2.3 Evaluation results

In this section, we describe the implementations of CBS, TAS, and (multi-level) preemption and provide the associated resource utilization numbers are given. Specifically, we first describe the implementation of CBS and TAS (see Section 3.2.3.1) and discuss their resource utilization (see Section 3.2.3.2). We then describe the implementation and resource utilization of multi-level preemption (see Section 3.2.3.3). The implementations themselves are proprietary and are therefore only presented at a high level of abstraction.

3.2.3.1 CBS & TAS Implementations

A functional block diagram of TAS and CBS is presented in Figure 3.4.

In this figure, the *configuration register bank* (separate for TAS and CBS) contains all the necessary run-time configurable registers and interface implementations so that both TAS and CBS modules can be managed by external software. The *CBS controller* implements the CBS algorithm and performs credit bookkeeping based on the current configuration and transmission status provided by the queuing system. CBS provides the *transmission allowed* flag for each supported priority, which is then used by the queuing system to select frames for transmission. For credit management, CBS needs information about the current *queue state* and *transmission status.*

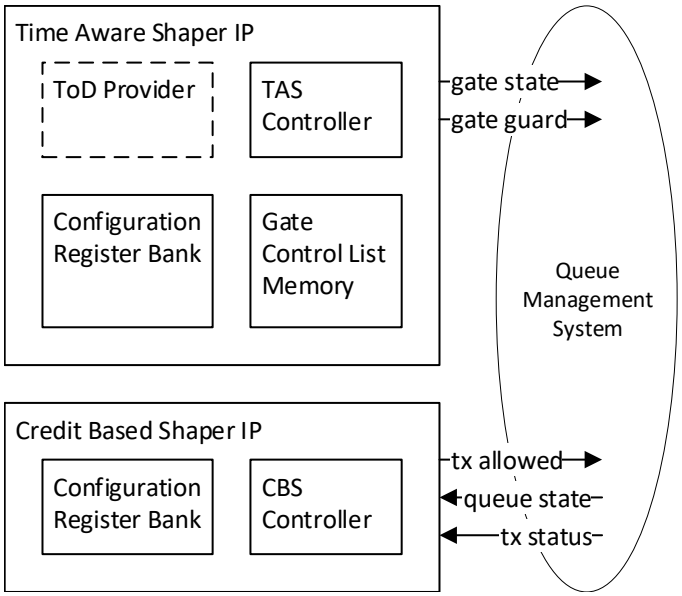


Figure 3.4: TAS and CBS block diagram

TAS provides the *gate state* and *gate guard* control signals to the queue management system. The *gate guard* control signal enables the use of fixed guard bands (as defined in Annex Q of IEEE 802.1Q (IEEE, 2018a) and in Section 2.4.4.1). This implementation of the guarding feature calculates how long the gate remains open, which is necessary to prevent transmission window overrun errors.

In addition, TAS contains a memory holding Gate Control Lists (GCL). By default, this memory is mapped to BRAMS. However, we can accurately estimate its size in bits based on the number of supported priorities and the number of supported entries. This estimate is shown in Equation 3.1, where M is the memory size in bits, n is the number of supported GCL entries, $\mathrm{TI}_{\mathrm{width}}$ is the bit size of the time interval entry, and N_{IPVs} is the number of supported priorities (Pruski et al., 2021).

$$M = 2 \times n \times \left(\mathrm{TI}_{\mathrm{width}} + N_{\mathrm{IPVs}}\right) \tag{3.1}$$

Finally, TAS requires a synchronized Time of Day (ToD) to be available in the system. This is illustrated by the *tod provider* entity in Figure 3.4. The overhead (in terms of resource usage) of this subsystem is included in our results for TAS and labeled as Time-Stamping Unit (TSU).

3.2.3.2 Resource utilization for TAS and CBS

For the TSN shapers, we performed synthesis runs for 1, 2, 4, and 8 supported priorities (N_IPVs). The corresponding counts for LUTs, Registers, CARRY8, and F7 Muxes are

shown in Figure 3.5.

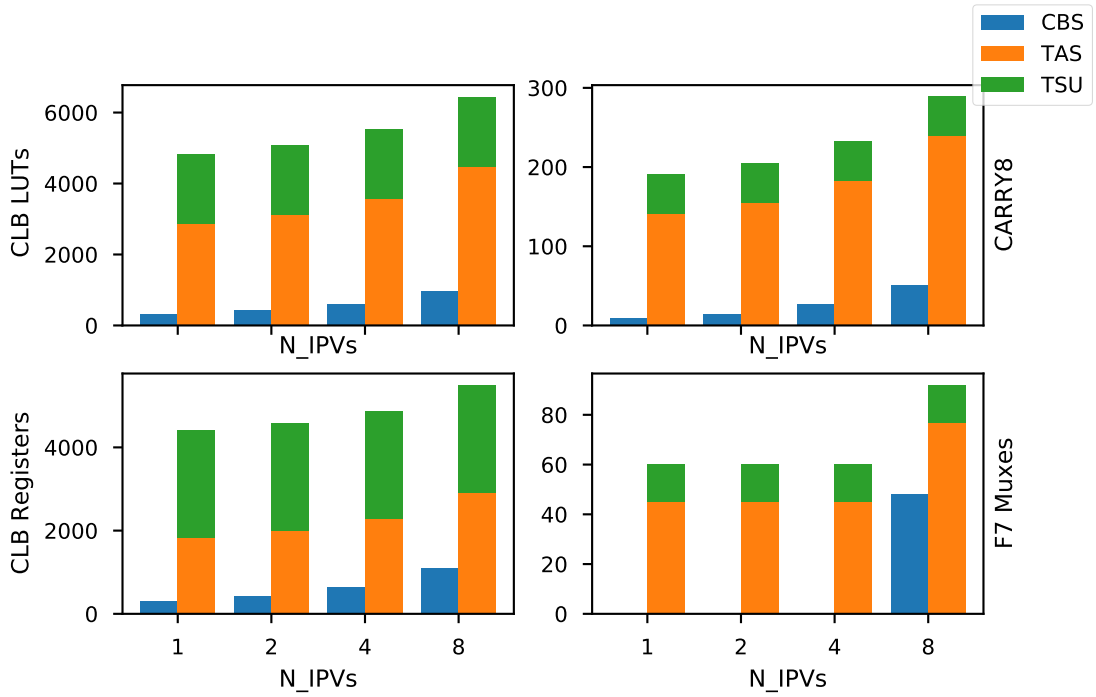


Figure 3.5: CBS and TAS CLBs

As can be seen, LUTs, Registers, and CARRY8 adders scale linearly with the number of supported priorities. This is explained by the fact that both the credit-keeping modules in CBS and the guarding modules in TAS are instantiated per priority. When TAS and CBS are used together, they affect each other in terms of resource utilization. However, we did not observe any significant additional overhead in such a scenario.

Table 3.1 provides the percentage of total resources available on `xczu9cg-ffvb1156-1-e` used by TAS IP (in rows marked *%TOT*), and also the overhead of TAS over CBS (how much more resources it uses, rows marked */CBS*). As expected, the TAS is significantly larger, with overhead being as high as 15 times larger in the case of only one supported priority. A significant portion of this can be attributed to the TSU (as seen in Figure 3.6), together with the previously explained guarding feature (not visualized). What is worth noticing, is that the LUT utilization of TAS is around 2%. With an instance of TAS required per egress port, it makes scaling it with port numbers especially expensive. Since F7 MUXes only start being used in CBS with 8 priorities, and their utilization numbers are very low, they are omitted from Table 3.1.

Table 3.1: TAS IP resource utilization and overhead over CBS

N_IPVs	Type	CLB LUTs	CLB Registers	CARRY8
1	%TOT	1.8%	0.8%	0.6%
	/CBS	1512%	1500%	2122%
2	%TOT	1.9%	0.8%	0.6%
	/CBS	1187%	1116%	1366%
4	%TOT	2%	0.9%	0.7%
	/CBS	913%	763%	863%
8	%TOT	2.4%	1%	0.8%
	/CBS	663%	500%	567%

3.2.3.3 Frame Preemption implementation

Recall that our proposed implementation approach of multi-level preemption is to introduce new pMAC interfaces on the MAC Merge Sublayer (MMS). However, we note that due to resource constraints (hardware availability and IP restrictions), we could only implement 1-level preemption feature on hardware and the hardware utilization results were extrapolated based on the predicted behavior of our implementation approach.

> We project that our approach causes a doubling of resources for the 2-level, a tripling for the 3-level, and so on.

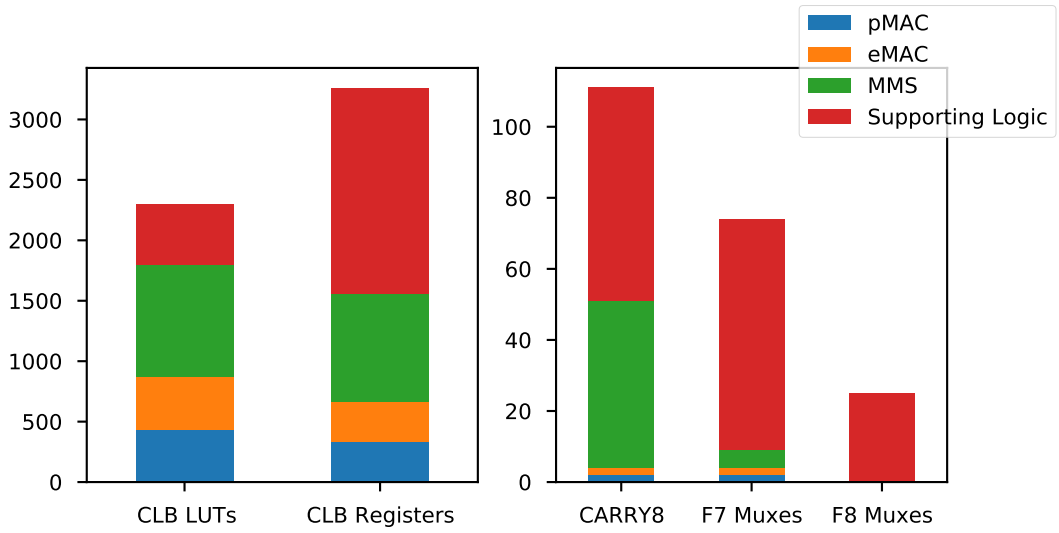


Figure 3.6: Resource utilization for TSN MAC

Table 3.2: Frame preemption resource utilization and overhead

Pre. levels	Type	CLB LUTs	CLB Registers	CARRY8	F7 Muxes
0	%TOT	0.16%	0.06%	0.01%	0.00%
1	%TOT	0.66%	0.28%	0.15%	0.01%
	%OH	411.21%	467.37%	2550.00%	450.00%
2	%TOT	1.15%	0.51%	0.29%	0.01%
	%OH	723.11%	835.03%	5000.00%	800.00%

In Figure 3.6, stacked bar charts are used to visualize the resource utilization of the major submodules of the TSN MAC. Table 3.2 shows the percentage of total resources available on the target device in *% TOT* rows. Since the increase in utilization is linear, only the resource utilizations for levels zero, one, and two are shown. In addition, in rows marked *% OH* list the implementation overhead compared to the non-preemption MAC (row 0). The values are exclusive of supporting logic. The MMS implements more complex preemption operations, which explains why its size is more than twice that of the simple MAC. With three traffic classes, a 2-level preemption scheme would be preferable to TAS if it can meet the performance requirements. However, if more traffic classes need to be supported, the choice is not so clear.

If multi-level preemption is implemented according to our specifications, the cost grows linearly. It is less than TAS for up to four levels of preemption, after which it overtakes TAS.

However, with a linear increase to eight classes, multi-level preemption would be significantly more expensive than TAS. This does not take into account the additional software required for time synchronization in TAS (TSU).

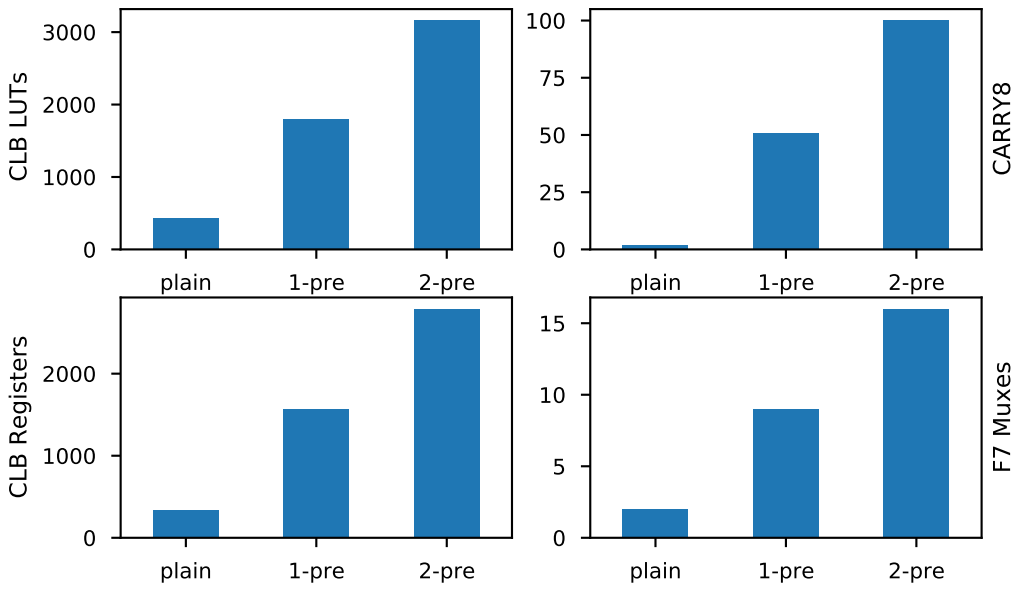


Figure 3.7: Resource increase caused by adding preemption levels.

Chapter Summary

In this chapter, we examine the transmission and reception functions as currently defined in the standards, and point out necessary changes to allow for an additional level of preemption. We then provide recommendations for interoperability and frame buffering. We also compare the implementation costs of frame preemption with those of TAS and CBS, based on FPGA resource utilization for each of the features. Based on our results, we find that TAS has the highest initial resource utilization among the three features, thereby making it the most expensive to implement. This is even more true when the cost of time synchronization is included. For Frame Preemption, the cost is significantly higher than that of CBS but much lower than that of TAS. When multi-level preemption is implemented according to our specifications, the cost increases linearly. Up to four levels of preemption, they are lower than those of TAS, after which they overtake TAS. We note that this projection does not include the additional overhead of time synchronization software required by TAS. CBS is the cheapest to implement, although it is not able to meet the stringent timing requirements of emerging applications.

Chapter 4

Model, Analysis, and Configuration

Meeting timing requirements is essential for any real-time communication medium. Therefore, formal timing analysis must be performed to provide safe temporal guarantees for frame transmission times when a new feature/scheme is proposed. In this chapter, we present results on the worst-case traversal time (WCTT) of frames under the assumption of multi-level preemption. As in most, if not all, real-time and/or time-critical preemptive systems, an appropriate priority allocation policy plays a central role in the resulting performance of both 1-level and multi-level preemption schemes to avoid over-provisioning and/or sub-optimal utilization of hardware resources. In addition, multi-level preemption raises new configuration issues. In particular, the correct number of preemption levels to enable, and the synthesis of the assignment of flows to preemption classes remain open problems. In this chapter, we address these configuration challenges by providing a new configuration framework. The remainder of the chapter is organized as follows. First, in Section 4.1, we introduce the system model and enforce a set of rules for network and traffic management. Then we present the WCTT analysis and the configuration framework in Sections 4.2 and 4.3, respectively.

4.1 Model

In this section, we introduce all the parameters and assumptions that we will use in this chapter. In particular, we introduce the network, traffic, and configuration specifications in Section 4.1.1, Section 4.1.2, and Section 4.1.3. We assume that all timing characteristics are non-negative integers, i.e., they are multiples of a discrete time interval (e.g., the CPU tick).

4.1.1 Network specification.

We assume an Ethernet backbone network for a distributed real-time embedded system. We represent the network as a directed graph $G \stackrel{\text{def}}{=} (\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ is the set of all nodes and $\mathcal{L}$ is the set of all physical links in the network. We assume that each link is bidirectional, full-duplex, and operates at a single reference speed, say $s > 0$. The tuple G is given with the interpretation that: (1) $\mathcal{N} = \text{EP} \cup \text{SW}$, where $\text{EP} \stackrel{\text{def}}{=} \{\text{EP}_1, \text{EP}_2, \ldots\}$ represents the set of all endpoints and $\text{SW} \stackrel{\text{def}}{=} \{\text{SW}_1, \text{SW}_2, \ldots\}$ is the set of all switches. Each EP_q (with $q \geq 1$) has a single input/output port and can receive and send network traffic, while SW consists only of forwarding nodes, each with a finite number of input/output ports over which traffic is routed. Each SW_ℓ (with $\ell \geq 1$) is equipped with a multi-level preemption capability and decides to which output port to forward a received frame based on its internal routing table. We also assume that each input/output port of the switch has 8 priority queues and that each queue has flows of the appropriate priority level assigned to it. Finally, we assume that each SW_ℓ supports multi-level preemption and Strict Priority (ST) transmission.

> According to the IEEE 802.1 Qbu standard, frame preemption can be implemented with or without the TSN shapers such as TAS and CBS. In this work, we choose the latter scenario (i.e., an implementation without shapers) to focus solely on evaluating multi-level preemption without the added complexity of other feature mechanisms.

4.1.2 Traffic specification.

We consider $\mathcal{F} \stackrel{\text{def}}{=} \{f_1, f_2, ..., f_n\}$ a network traffic with $n \geq 1$ flows. Each flow $f_i \stackrel{\text{def}}{=} \langle src_i, dst_i, T_i, D_i, S_i, P_i, \text{PC}_i \rangle$ consists of a potentially infinite number of instances (a.k.a. frames) and is characterized by: (1) src_i, the source endpoint; (2) dst_i, the destination endpoint; (3) T_i, the minimum inter-arrival time between two consecutive frames of f_i, i.e., assuming that the first frame of f_i is released at src_i at time $a_{i,1} \geq 0$, then $a_{i,\lambda+1} - a_{i,\lambda} \geq T_i$ for all $\lambda \geq 1$, where $a_{i,\lambda}$ is the release time of the λ^{th} frame; (4) D_i, the relative deadline, i.e., $d_{i,\lambda} \stackrel{\text{def}}{=} a_{i,\lambda} + D_i$ is the latest time at which the λ^{th} frame of f_i must reach dst_i; (5) S_i, the size of f_i (in bytes); (6) P_i (with $0 \leq P_i \leq 7$), the priority; and finally (7) PC_i the preemption class. For simplicity, we assume that 0 is the highest priority and 7 is the lowest. The specification in the standards suggests otherwise. However, we have chosen to keep both the frame priority and preemption class in the same format (ascending order, starting with 0) to improve readability. For the preemption classes, we also assume that the smaller the

value, the higher the preemption class. Flows with the same priority always belong to the same preemption class, but the converse is not true. In other words, flows with the same preemption class can have different priorities (see Figure 4.1).

4.1.3 Flow configuration.

For a flow set $\mathcal{F}$, we denote by $\mathcal{C}^m \stackrel{\text{def}}{=} \{\mathcal{C}^m_1, \mathcal{C}^m_2, \mathcal{C}^m_3, \ldots\}$ the set of all possible flow configurations under the assumption of an m-level preemption scheme (with $0 \leq m \leq 7$). We recall that an m-level preemption scheme implies $m+1$ preemption classes. Each configuration $\mathcal{C}^m_x$ (with $x \geq 1$) is an *integer list* defining the preemption class of each priority level. Specifically, $\mathcal{C}^m_x is \stackrel{\text{def}}{=} [c^m_{x,0}, c^m_{x,1}, \ldots, c^m_{x,p-1}]$, where p is the number of distinct priorities in $\mathcal{F}$ and $c^m_{x,\sigma}$ (with $0 \leq \sigma \leq p-1$) is the preemption class of all flows $f_i \in \mathcal{F}$ with priority $P_i = \sigma$. In other words, a preemption configuration is a nondecreasing and surjective function. Figure 4.1 shows an example of a flow configuration $\mathcal{C}^3_x$ for a 3-level preemption scheme (i.e., with 4 preemption classes).

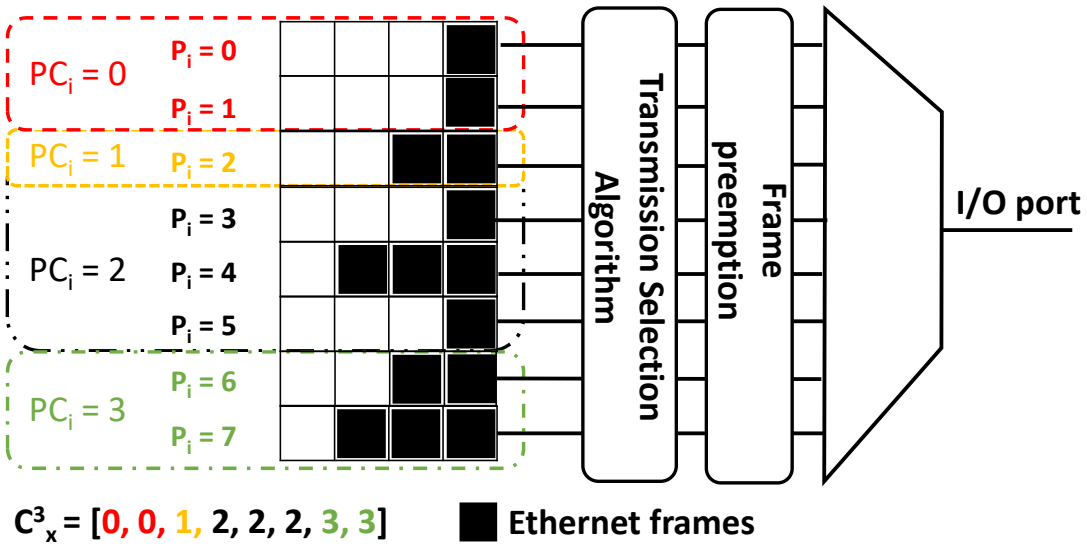


Figure 4.1: Preemption classes; priority queues and Configuration.

In this figure, we assume that $x = 1$ to refer to the "first configuration" from the set of all possible configurations $\mathcal{C}^3$. Flows with priorities 0 and 1 are assigned to the preemption class 0, i.e., $c^3_{1,0} = 0$ and $c^3_{1,1} = 0$; flows with priority 2 are assigned to preemption class 1; flows with priorities 3, 4 and 5 are assigned to preemption class 2; and finally, flows with priorities 6 and 7 are assigned to preemption class 3. Thus, the resulting configuration is $\mathcal{C}^3_1 = [0,0,1,2,2,2,3,3]$. Note that we assume a network-wide configuration, i.e., $\mathcal{C}^3_1$

applies to all switches. Also, for any m-level preemption scheme, the following rules apply.

R_1– Each flow f_i can be assigned to one and only one preemption class;

R_2– To reduce priority inversion, a flow, say f_j, with a lower priority than another flow, say f_i, cannot be assigned to a higher preemption class than that of f_i;

R_3– To conserve hardware resources, at least one flow must be assigned to each preemption class.

R_4– Any flow, say f_i, can preempt any other flow, say f_j, only if $\mathrm{PC}_i < \mathrm{PC}_j$, but flows in PC_j cannot preempt flows in PC_i;

R_5– Following the TSN standard convention, flows in the same preemption class cannot preempt each other and are transmitted in a strict priority order;

R_6– Flows with the same priority are transmitted in a FIFO manner;

R_7– Flows with the same priority always belong to the same preemption class, but the converse is not true. In other words, flows with different priorities can be assigned to the same preemption class.

For ease of reading, an overview of the notations used in this chapter is provided in Table 4.1.

Table 4.1: Overview of key variables

a_i^q	Arrival time of the q^{th} frame (f_i^q) of flow f_i
$\mathcal{BP}$	Set of best-effort preemptable flows
C_i^+	Maximum time to transmit any frame of f_i
PC_i	Set of flows in the same preemption class as f_i
$\delta_i^{+/-}$(q)	Latest/earliest arrival time of f_i^q
$\mathcal{E}$	Set of eMAC flows
F_i^+	Maximum number of fragments of C_i^+
$hep(i)$	Set of flows with a higher priority than or equal to f_i
HPI_i	Higher-priority interference suffered by f_i
$hp(i)$	Set of flows with a higher priority than f_i
$lp(i)$	Set of flows with a lower priority than f_i
LPB_i	Lower-priority blocking suffered by f_i (including express frames)
mWCTT	Maximum measured end-to-end delay
$\eta_i^{+/-}(\Delta t)$	Maximum/Minimum possible arrivals of a frame of f_i within the period Δt
$p_i^{+/-}$	Maximum/Minimum payload of frames of f_i
PB_i	Blocking suffered by f_i due to the transmission of a lower-priority preemptable frame/frame fragment
$PO_i(\Delta t, q, a_i^q)$	Preemption overhead incurred by f_i^q
$Q_i(q, a_i^q)$	Queuing delay of f_i^q
rTX	Link data transmission rate
SPB_i	Same-priority blocking suffered by f_i
$sp(i)$	Set of all flows with the same priority as f_i
$\mathcal{TP}$	Set of time-sensitive preemptable flows
$\mathrm{WCTT}(q, a_i^q)$	Worst-case traversal time of f_i^q

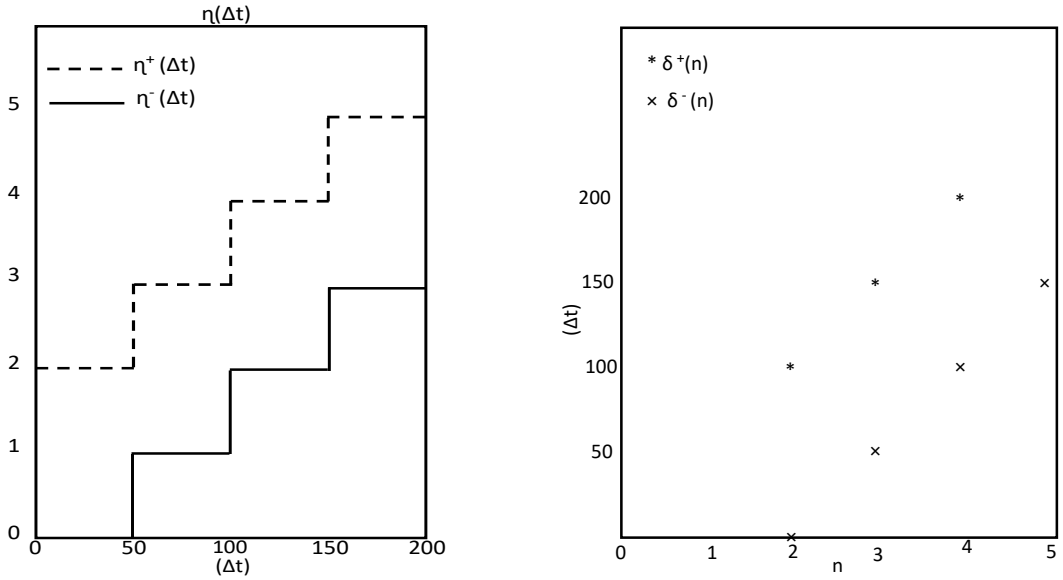
4.2 Analysis

In this section, we present results on the worst-case traversal time (WCTT) of frames under the assumption of multi-level preemption. To this end, we extend the worst-case performance analysis for 1-level preemption presented by Thiele and Ernst (2016a), where the authors used the Compositional Performance Analysis (CPA) framework to analyze the delay experienced by each frame in a switched Ethernet network. We recall that CPA is useful for analyzing large and complex systems because it allows decomposing the performance of a system into the performance of its individual components, which allows for identifying bottlenecks and optimizing the overall performance of the system. First, we give a brief background on CPA in Section 4.2.1 and then present the analysis in Section 4.2.2

4.2.1 A brief background on CPA

Before presenting our proposed WCTT analysis, it is important to give the reader a brief background on the CPA approach so that they can easily understand the rest of this chapter. For a complete and detailed description of the basic concepts, we refer the interested reader to (Henia et al., 2005). In this framework, a component is modeled as a resource ρ that provides a service to one or more tasks. Task activations are abstracted by an event model that defines an upper-bound (denoted by $\eta^+(\Delta t)$) and a lower-bound (denoted by $\eta^-(\Delta t)$) on the number of activations within a half-open time interval $[t, t+\Delta t)$. This framework also defines a distance function $\delta^+(q)$ (resp. $\delta^-(q)$), which gives an upper (resp. lower) bound on the maximum (resp. minimum) time at which the activation instance q^{th} can occur (Henia et al., 2005; Hofmann et al., 2017). Figure 4.2 illustrates the event model of a task with a period and jitter of 50-time units. A jitter of 50 time units (and equal to the period) means that two instances can occur simultaneously ($\delta^-(2) = 0$; $\eta^+(0) = 2$) or two periods apart ($\delta^+(2) = 100$; $\eta^-(100) = 0$).

The service period of each task, say τ_i, is bounded by two parameters: from above by C_i^+ and from below by C_i^-. These parameters represent the longest and shortest time, respectively, that the resource, say ρ, takes to service an instance of τ_i without blocking or interference. The worst-case response time (WCRT) of the task τ_i is examined within what is called a *level-i busy period*. In summary, this is a time interval $[\sigma, \mu]$ in which only task instances belonging to $hep(i)$ (except the first job generated from task $\tau_j \in lp(i)$ with the longest C_j^+), executed throughout $[\sigma, \mu]$, but no jobs belonging to $hep(i)$ are executed in $[\sigma - \varepsilon, \sigma)$ or $(\mu, \mu + \varepsilon]$ for any arbitrary small $\varepsilon > 0$.



(a.) An event arrival function. (b.) An event distance function

Figure 4.2: CPA event model for a task with a period and jitter of 50 time units.

Since the WCRT of τ_i may not occur on its first activation, the CPA examines all q activations of τ_i within the level-i busy period. To this end, it defines an *q-activation processing time* $B_i(q)$, which describes how long the resource ρ is busy processing q jobs of τ_i. This is computed in Equation 4.1.

$$B_i(q) = Q_i^+(q) + C_i^+ \tag{4.1}$$

Note in Equation 4.1 that $B_i(q)$ assumes a non-preemptive scheduling scheme. Here $Q_i^+(q)$ is the time interval between the start of the level-i busy period and the start of service of the q^{th} instance of τ_i. It is calculated as in Equation 4.2.

$$Q_i^+(q) = \max_{j \in lp(i)} \left\{ C_j^+ \right\} + (q-1) \cdot C_i^+ + \sum_{k \in hp(i)} C_k^+ \cdot \eta_k^+ \left(Q_i(q) + \varepsilon \right) \tag{4.2}$$

In Equation 4.2, $lp(i)$ and $hp(i)$ denote the set of tasks with a lower and higher priority than τ_i, respectively. The first term computes the maximum delay that can be caused by the presence of a lower-priority task in the system; the second term estimates the delay caused by processing all previous $q-1$ instances, and the third term computes all possible delays caused by serving higher priority tasks within $B_i(q)$. It is worth noting that Equation 4.2 is reminiscent of the Worst-Case Response Time estimation for the classical single-core fixed-priority tasks and is also solved using a fixed-point algorithm, i.e., the solution is computed iteratively and the algorithm stops as soon as two consecutive values

of $Q_i^+(q)$ are equal or when a value exceeds the deadline. In the latter case, the set of tasks is deemed unschedulable. Having obtained $Q_i^+(q)$, we can derive the maximum number of activations q_i^+ of τ_i within $B_i(q)$ using Equation 4.3.

$$q_i^+ = \min\{q \geq 1 \mid B_i(q) < \delta_i^-(q+1)\} \tag{4.3}$$

This equation computes the first q_i instances of τ_i such that the completion time of instance q_i is less than the earliest arrival time of instance q_{i+1}. The response time of the q^{th} instance of τ_i is given by the difference between the time when all q instances of τ_i are processed and the arrival time of instance q. Formally, this is given by Equation 4.4.

$$R_i^+(q) = B_i(q) - \delta_i^-(q) \tag{4.4}$$

Finally, the WCRT R_i^+ of τ_i is the maximum $R_i^+(q)$ over all q_i^+ activations of τ_i within the busy window and is calculated by Equation 4.5.

$$R_i^+ = \max_{1 \leq q \leq q_i^+} \{R_i^+(q)\} \tag{4.5}$$

So far, we have discussed the worst-case delay at each resource node, also referred to as *local analysis*. Specifically, at each resource node, a rigorous computation is performed on the maximum possible blocking/interference that a task can suffer. This means that the potential interference between any two tasks is effectively captured when performing local analysis at the shared resource node. However, in many cases, real-time applications involve interacting tasks that share different resources. To evaluate the WCRT of a task in this scenario, the CPA defines an *event propagation* step in addition to the local analysis step. Here, the output of each resource along the execution path of a task serves as the input to the next, and $R_i^+(q)$ at the last node is the WCRT of the task.

The CPA approach has been applied to standard Ethernet (Rox and Ernst, 2010; Thiele et al., 2015b), AVB (Diemer et al., 2012a), and TSN (Thiele and Ernst, 2016b,a). In these works, the output ports of the switches are *resources* and flows are *tasks*. The frames are *jobs* or *instances* of the tasks and the path of a flow is modeled as a chain of dependent tasks (Diemer et al., 2012a). Specifically, resources render some *service(s)* (transmission) to some task activations (frames) upon the occurrence of an event (arrival of frames) within a time window (a level-i busy period). In this context, the service period of each frame is bounded by two parameters: from above by C^+ and from below by C^-. These parameters represent the longest and the shortest time for the frame to transmit in the absence of any interference. They depend, of course, on the minimum and maximum possible payload $p^{-/+}$ of the flow to which the frame belongs and on the speed of the

output link. Equation 4.6 illustrates the relationship between these parameters[1].

$$C^{-/+} = \frac{42\,\text{bytes} + \max\{42\,\text{bytes},\, p^{-/+}\}}{rTX} \tag{4.6}$$

In this equation, rTX represents the transmission speed on the port link and the constant terms (here, 42) represent the protocol overhead and minimum frame payload requirement as specified in the Standards (IEEE, 2018a).

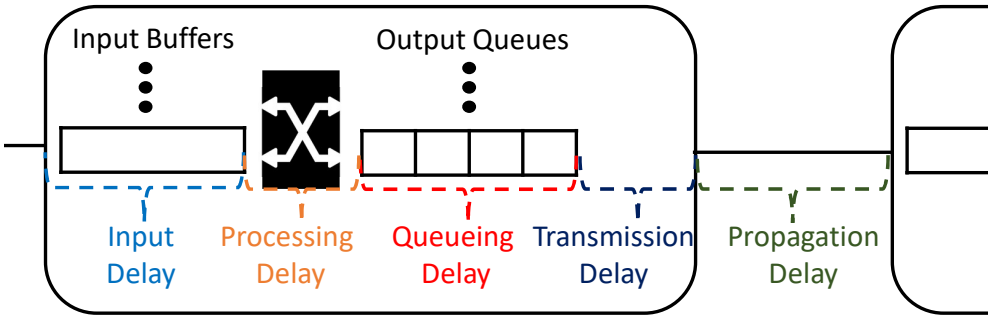


Figure 4.3: End-to-end delay components.

As shown in Figure 4.3, the end-to-end delay experienced by a frame in a switch fabric is composed of the following five components: (1) the input delay at the switch input port; (2) the processing delay; (3) the queuing delay at the switch output port; (4) the propagation delay; and finally (5) the transmission delay (Finn, 2017). As pointed out by Thiele and Ernst (2016a), all of these components are implementation-dependent and are generally on the order of a few clock cycles, with the exception of the queueing delay, which is captured in Equation 4.2 and whose components are shown in Figure 4.4.

In Figure 4.4, frame f_i^q arrives at time a_i^q during the transmission of a frame of the lower preemption class (green box). Before a_i^q, the express class frame (first red box) and two other frames (first two yellow boxes) with the same priority and preemption class as f_i^q had arrived. The preemption operation commands that a minimum number of bytes have been transmitted and a number of bytes remain for the preempted fragment to satisfy the minimum size of a valid Ethernet frame. This brings the highest possible blocking that a preempting frame can experience to the size of the largest non-preemptable fragment of a preemptable frame, i.e., 143 bytes of data (Thiele and Ernst, 2016a) and explains the transmission process of the lower priority frame before it is preempted. After the lower priority frame is preempted, the express frame, all pending frames of the same priority,

[1]The minimum size of a frame is 84 bytes according to the specification of the Standard

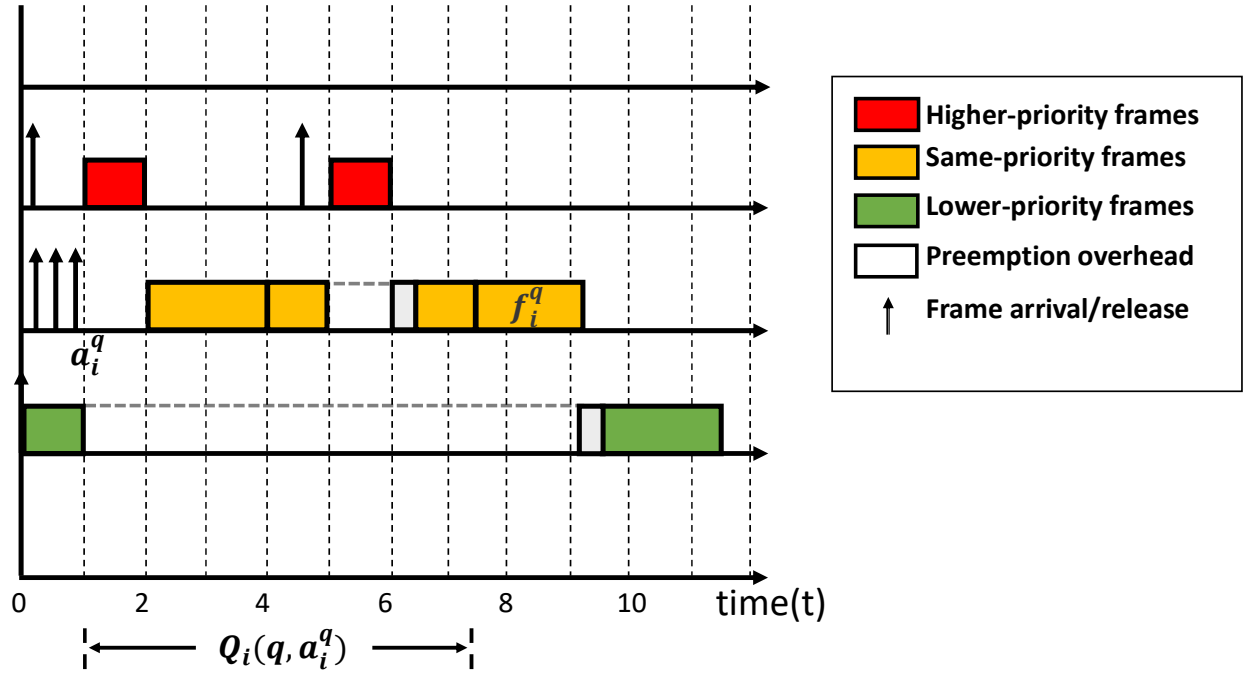


Figure 4.4: Queuing delay components.

and all newly arrived express frames are transmitted, thus forcing f_i^q to be served only afterward.

In addition to the queueing delay components captured in Equation 4.2, preemption overhead is another term that should be considered. This is not the case in the general CPA model (Hofmann et al., 2017). Preemption overhead is an important factor in the TSN frame transmission scheme. The total overhead incurred by each preemption is 12 bytes (i.e., 6-byte preamble, 1-byte start frame delimiter, 1-byte frame count variable; and finally 4-byte error check variable) (Ojewale et al., 2018). In addition, the *Inter Frame Gap* (IFG) between two consecutive transmissions must be considered before the next frame/fragment is transmitted. According to the standards, the size of each IFG is equal to the time required to transmit 12 bytes of data. Thus, the total overhead associated with each preemption amounts to 24 bytes. Thiele and Ernst (2016a) have also proved that the maximum number of preemptions that a single frame can suffer is given by Equation 4.7.

$$F_i^+ = \left\lfloor \frac{p_i^+ - 42\,bytes}{60\,bytes} \right\rfloor \tag{4.7}$$

Equation 4.7 provides an upper bound on the number of times an Ethernet frame can be preempted based on the minimum valid frame size constraints defined in the standards. In this equation, the constant terms (here 42 and 60) are the minimum payload requirements for the first and subsequent fragments of a preemptable frame in the stan-

dards (IEEE, 2018a). For each flow f_i, its worst-case traversal time is obtained by combining the Equations 4.1 and 4.4.

We note that Equation 4.4 does not account for preemption overhead. This computation is refined to include the cost in Section 4.2.2.6. Note also that the CPA model allows for arbitrary deadlines, i.e., there is no particular correlation between the deadline and the minimum inter-arrival time of each flow.

4.2.2 Proposed analysis

In the multi-level preemption scheme, each frame can belong to only one of the following three categories: (1) can preempt other frames and is not preemptable, i.e., it cannot be preempted by any other frame because it belongs to the highest preemption class (e.g., express traffic); (2) can preempt other frames and can also be preempted; and finally, (3) cannot preempt other frames and can be preempted by any frame in the categories (1) and (2). We refer to the categories (1), (2), and (3) as "express flows", "preemptable flows with firm timing requirements" (tpflows), and "best effort flows" (bpflows), respectively. With this nomenclature in mind, we have all we need to discuss the components of the end-to-end delay of a frame, say f_i^q, arriving at time a_i^q. Briefly, there are four components, namely:

1. the *lower priority blocking* – i.e., the delay due to frames with a lower priority than f_i;
2. the *same-priority blocking* – i.e., the delay due to frames with the same priority as f_i;
3. the *higher priority interference* – i.e., the delay associated with frame(s) with a higher priority than f_i; and finally,
4. the *preemption overheads*, i.e., the delay due to preemption overheads.

4.2.2.1 Lower-priority blocking

In this section, we derive an upper bound on the lower-priority blocking experienced by a frame in each flow class.

▷ **Lower-priority blocking for "express flows".** Each express frame can experience the maximum lower-priority blocking in two scenarios: (*i*) if it is blocked by the largest

lower-priority express frame since frames in the same preemption class are served non-preemptively; or (*ii*) if it is blocked by the largest non-preemptable fragment of a preemptable frame. Recall that any preemptable frame less than or equal to 143 bytes is non-preemptable. Lemma 1 provides an upper bound for blocking due to scenario (*ii*).

Lemma 1 ($\mathrm{PB}_i^{\mathcal{E}}$)**.** *For any express flow $f_i \in \mathcal{E}$, the maximum blocking of a frame f_i^q (with $q \geq 1$) caused by a preemptable frame is given by Equation 4.8.*

$$\mathrm{PB}_i^{\mathcal{E}} = \min \left\{ \underbrace{\max_{j \in \mathcal{P}} \left\{ C_j^+ \right\}}_{(a)}, \underbrace{\frac{143\,bytes}{rTX}}_{(b)} \right\} \tag{4.8}$$

Proof. If all preemptable frames are shorter than 143 *bytes*, then the maximum blocking time for any express frame is caused by the largest of these frames. This is captured by term (*a*). On the other hand, the longest non-preemptable fragment of any preemptable frame is 143 bytes long (Thiele and Ernst, 2016a). This implies a maximum blocking time of $\frac{143\,bytes}{rTX}$ (where rTX is the link speed). This is captured by term (*b*). Therefore, a tight upper bound on the blocking time caused by a preemptable frame to any express frame f_i^q (with $q \geq 1$) is given by the minimum between terms (*a*) and (*b*), and the lemma follows. □

From Lemma 1, Theorem 1 provides a tight upper bound on the lower-priority blocking incurred by any express frame.

Theorem 1 ($\mathrm{LPB}_i^{\mathcal{E}}$)**.** *For any express flow $f_i \in \mathcal{E}$, if $lp^{\mathcal{E}}(i)$ represents the set of express frames with a lower priority than that of f_i, then a tight upper bound on lower-priority blocking for each frame f_i^q (with $q \geq 1$) is given by Equation 4.9.*

$$\mathrm{LPB}_i^{\mathcal{E}} = \max \left\{ \underbrace{\max_{j \in lp^{\mathcal{E}}(i)} \left\{ C_j^+ \right\}}_{(a)}, \underbrace{\mathrm{PB}_i^{\mathcal{E}}}_{(b)} \right\} \tag{4.9}$$

Proof. Each express frame f_i^q can suffer lower-priority blocking due either to the transmission of (1) a lower-priority express frame or (2) preemptable frame. In the first case, term (*a*) captures the largest blocking time, since frames of the same preemption class are served non-preemptively. On the other hand, if the blocking is caused by a preemptable frame, then Lemma 1 provides an upper bound for lower-priority blocking, i.e., $\mathrm{PB}_i^{\mathcal{E}}$ (term (*b*)). Therefore, a tight upper bound on the blocking time of any express frame f_i^q (with $q \geq 1$) is given by the maximum between terms (*a*) and (*b*), and the theorem follows. □

Note that Equations 4.8 and 4.9 are similar to Equations 5 and 7 in (Thiele and Ernst, 2016a). We explicitly address this behavior in the analysis for the sake of completeness.

▷ **Lower-priority blocking for "tpflows".** Any flow can preempt all flows in a lower preemption class by design, but the converse is not true. A tpflow is not an exception to this rule. This means that any tpflow frame f_i^q can be blocked at most by (*i*) the largest non-preemptable fragment of any preemptable frame in a lower preemption class or (*ii*) a lower-priority frame of the same preemption class as f_i, since frames of the same class are serviced in a non-preemptable way. Lemma 2 computes an upper bound on the blocking time incurred by a tpflow frame due to frames in lower preemption classes.

Lemma 2 ($\text{PB}_i^{\mathcal{TP}}$)**.** *For each flow $f_i \in \mathcal{TP}$, the maximum blocking time of each frame f_i^q (with $q \geq 1$) caused by a frame of a lower preemption class is given by Equation 4.10.*

$$\text{PB}_i^{\mathcal{TP}} = \min \left\{ \underbrace{\max_{j \in \{lp(i) \mid \text{PC}_j > \text{PC}_i\}} \left\{ C_j^+ \right\}}_{(a)}, \underbrace{\frac{143\,bytes}{rTX}}_{(b)} \right\} \tag{4.10}$$

Proof. The proof of Lemma 2 is similar to that of Lemma 1. Given a frame f_i^q, if all frames in a lower preemption class are shorter than 143 *bytes*, then the maximum blocking incurred by f_i^q is caused by the largest of these frames. This is captured by term (*a*). Otherwise, the longest non-preemptable fragment of any preemptable frame is 143 bytes long (Thiele and Ernst, 2016a) and this implies a maximum blocking time of $\frac{143\,bytes}{rTX}$ (where rTX is the link speed). This is captured by term (*b*). Therefore, a tight upper bound on the blocking time caused by a preemptable frame of a preemption class lower than f_i^q is given by the minimum between terms (*a*) and (*b*), and the lemma follows. □

From Lemma 2, Theorem 2 derives a tight upper bound on the lower-priority blocking that occurs on any tpflow frame.

Theorem 2 ($\text{LPB}_i^{\mathcal{TP}}$)**.** *For any tpflow flow $f_i \in \mathcal{TP}$, the maximum lower-priority blocking of any frame f_i^q (with $q \geq 1$) is given by Equation 4.11.*

$$\text{LPB}_i^{\mathcal{TP}} = \max \left\{ \underbrace{\max_{j \in \{lp(i) \mid \text{PC}_i = \text{PC}_j\}} \left\{ C_j^+ \right\}}_{(a)}, \underbrace{\text{PB}_i^{\mathcal{TP}}}_{(b)} \right\} \tag{4.11}$$

Proof. Each tpflow frame f_i^q can be blocked either due to the transmission of (1) frame with lower priority of the same preemption class or (2) a frame with a lower preemption

class. In the first case, term (a) captures the largest blocking time that can be caused by a frame of the same preemption class, since these frames are served in a non-preemptive manner by design. In the second case, when the blocking is caused by a frame of a lower preemption class, we have already shown in Lemma 2 that this delay cannot exceed $\mathrm{PB}_i^{\mathcal{TP}}$ (i.e., term (b)). Therefore, a safe upper bound on the blocking time suffered by any tpflow frame f_i^q (with $q \geq 1$) is given by the maximum between terms (a) and (b), and the theorem follows. □

▷ **Lower-priority blocking for "bpflows".** Since bpflows by assumption belong to the lowest preemption class, it follows that the maximum lower-priority blocking that a bpflow frame can experience is given by the largest lower-priority frame in the same class. This is given by the Equation 4.12.

$$\mathrm{LPB}_i^{\mathcal{BP}} = \max_{j \in lp_{(i)}^{\mathcal{BP}}} \left\{ C_j^+ \right\} \tag{4.12}$$

Now that we have discussed in detail the computation of all lower-priority blocking terms, we can proceed with the computation of the same-priority blocking – i.e., the delay due to frame(s) with the same priority as the flow under analysis.

4.2.2.2 Same-priority blocking

In the following subsections, we compute upper bounds on the delay that a frame experiences when frames of the same priority are transmitted.

▷ **Same-priority blocking for "express flows".** Each express frame f_i^q can be blocked by all frames of the same priority that arrive before it at the time, say a_i^q, within the level i busy period. Moreover, all previous $q-1$ instances of f_i must be transmitted before f_i^q is transmitted, and therefore contribute to the same priority blocking term. Therefore, the maximum same-priority blocking $\mathrm{SPB}_i^{\mathcal{E}}$ that express frame f_i^q can experience is given by Equation 4.13.

$$\mathrm{SPB}_i^{\mathcal{E}}(q, a_i^q) = \sum_{j \in sp(i)} \eta_j^+(a_i^q) \cdot C_j^+ + (q-1) \cdot C_i^+ \tag{4.13}$$

In Equation 4.13, $sp(i)$ denotes the set of all flows with the same priority as f_i. The first term computes the maximum delay due to the transmission of all frames with the same priority that arrive before a_i^q, while the second term computes the maximum delay that f_i incurs due to the transmission of all previous $q-1$ instances.

▷ **Same-priority blocking for "tpflow".** Similar to express frames, each tpflow frame f_i^q can be blocked by all frames with the same priority that arrive before a_i^q within the

level-i busy period, as well as by all previous $q-1$ instances of f_i. Moreover, since f_i^q is preemptable, there may be an additional delay even after its transmission begins. Only during the transmission of its last fragment, which is not preemptable, is it guaranteed not to be interrupted. In other words, the last fragment of f_i^q must wait until all previous fragments of the instance have been transmitted. The size of this last fragment is equal to the minimum valid Ethernet frame size (Thiele and Ernst, 2016a). With the above, the maximum same-priority blocking $\text{SPB}_i^{\mathcal{TP}}$ that f_i^q can suffer is given by Equation 4.14.

$$\text{SPB}_i^{\mathcal{TP}}(q, a_i^q) = \sum_{j \in sp(i)} \eta_j^+(a_i^q) \cdot C_j^+ + (q-1) \cdot C_i^+ + \left(C_i^+ - \frac{84\,bytes}{rTX} \right) \tag{4.14}$$

In Equation 4.14, the first term computes the maximum delay due to the transmission of all frames with the same priority as f_i that arrive before a_i^q, while the second term computes the delay due to the transmission of all previous $q-1$ instances of f_i. Finally, the third term computes the maximum delay due to the transmission of all non-final fragments of f_i^q.

▷ **Same-priority blocking of "bpflows".** The notion of same-priority blocking for bpflows is identical to that of tpflows in Equation 4.14. The reason for this is that the behavior of frames within these classes is identical since they are transmitted non-preemptively and in a FIFO manner. This means that a bpflow frame f_i^q can be blocked by all frames with the same priority that arrive before its arrival time a_i^q within the level-i busy period; all previous $q-1$ instances of f_i; and by all preceding fragments (i.e., the non-terminal fragments) if it has been preempted. For this class, Equation 4.14 also suffices, with the $\mathcal{TP}$ superscript replaced by $\mathcal{BP}$ to reflect the corresponding frame class.

4.2.2.3 Higher-priority Interference

Regardless of its class, any frame f_i cannot start its transmission if another frame with higher priority (Thiele et al., 2014) is present. This means that any frames of higher priority that arrive before the frame f_i^q is transmitted will always interfere with its transmission. It follows that the higher-priority interference term for the three classes of frames is given by Equation 4.15.

$$\text{HPI}_i(\Delta t) = \sum_{j \in hp(i)} \eta_j^+(\Delta t) \cdot C_j^+ \tag{4.15}$$

4.2.2.4 Preemption overheads

Each preemption operation has some overhead associated with it, equal to the time it takes to transmit 24 bytes of data. This overhead is always added to the transmission time of the

preempted frame. Since express frames are transmitted in a non-preemptive manner, they do not incur preemption overhead. Therefore, only tpflows and bpflows incur preemption overheads since they are preemptable. From an analytical point of view, the approach to compute the preemption overhead terms is identical for these two preemption classes.

Roughly speaking, the maximum preemption overheads that a preemptable frame f_i^q can incur depends only on the *maximum number* of preemption events that can occur between its arrival time until the transmission of the first bit of the last non-preemptable fragment. In this sense, the maximum number of preemption events can occur in either of the following two cases:

(1) all preemptable frames transmitted between a_i^q and the complete transmission of f_i^q are preempted for the maximum number of times beyond which cutting a frame into more fragments would violate the minimum Ethernet frame size (see Equation 4.7).

(2) all possible arrivals of higher priority frames belonging to a higher preemption class occur and each of them causes a preemption.

To evaluate the first case, we consider a preemptable frame f_i^q transmitted within the busy period of length, say Δt. Then we compute the maximum number of times that f_i^q and all other preemptable frames transmitted within the window of size Δt can be preempted. We can distinguish between three different types of preemptable frames, namely: (1) preemptable frames with a lower priority than f_i; (2) preemptable frames with the same priority as f_i; and finally (3) preemptable frames with a higher priority than f_i. We discuss each of these three cases below.

▷ **Maximum number of preemptions incurred by a lower-priority preemptable frame.** This maximum number of preemptions occurs when a preemptable frame with lower priority, e.g. f_j^k, which is obstructing the transmission of f_i^q gets preempted a maximum number of times. Frame f_j^k can either (*i*) belong to a lower preemption class than f_i^q or (*ii*) have the same preemption class as f_i^q.

Case (*i*). If f_j^k belongs to a lower preemption class, then it will be preempted at most once by f_i^q or by another frame belonging to a higher class. By design, the preempted frame will not resume its transmission until all pending frames of a higher preemption class (including f_i^q) have completed their transmission.

Case (*ii*). If f_j^k is in the same preemption class as f_i^q, then f_j^k can be preempted multiple times and f_i^q can begin its transmission only after f_j^k has completed its transmission. Using Equation 4.7, we derive the maximum number of preemptions $N_i^{\mathcal{P},lp}$ incurred by

a preemptable frame with lower priority in the same preemption class as f_i^q in Equation 4.16.

$$N_i^{\mathcal{P},lp} = \max_{\{j \mid \text{PC}_i = \text{PC}_j \wedge i > j\}} \left\{ F_j^+ \right\} \tag{4.16}$$

▷ **Maximum number of preemptions incurred by same-priority preemptable frames.** This number includes the preemptions incurred by frames with the same priority as f_i^q that arrive before a_i^q within the level-i busy period as well as the preemptions suffered by f_i^q up to its last non-preemptable fragment. Using Equation 4.7, the maximum number of preemptions $N_i^{\mathcal{P},sp}(q, a_i^q)$ is computed in Equation 4.17.

$$N_i^{\mathcal{P},sp}(q, a_i^q) = q \cdot F_i^+ - 1 + \sum_{j \in sp(i)} \eta_j^+(a_i^q) \cdot F_j^+ \tag{4.17}$$

The first term of Equation 4.17 computes the maximum preemption incurred by the first q instances of f_i apart from the last fragment, while the second term computes the maximum preemption incurred by all other frames with the same priority as f_i arriving before f_i^q within the busy-window.

▷ **Maximum number of preemptions incurred by higher-priority preemptable frames.** This number includes the preemptions incurred by all frames with a higher priority than f_i^q transmitted during Δt, i.e., the transmission period of f_i^q. It is denoted by $N_i^{\mathcal{P},hp}(\Delta t)$ and occurs when all these frames are preempted as many times as possible, which is given by Equation 4.18.

$$N_i^{\mathcal{P},hp}(\Delta t) = \sum_{\{j \in hp(i) \wedge j \in \mathcal{P}\}} \eta_j^+(\Delta t) \cdot F_j^+ \tag{4.18}$$

Putting Equations 4.16, 4.17, and 4.18 together, the maximum number of preemptions incurred by the preemptable frames transmitted during Δt is obtained from Equation 4.19.

$$N_i(q, a_i^q, \Delta t) = N_i^{\mathcal{P},lp} + N_i^{\mathcal{P},sp}(q, a_i^q) + N_i^{\mathcal{P},hp}(\Delta t) \tag{4.19}$$

With this Equation 4.19 we have everything we need to derive an upper bound for the maximum preemption overhead of any preemptable frame f_i^q, as formalized in the Theorem 3 below.

Theorem 3 ($\text{PO}_i^{\mathcal{P}}(\Delta t, q, a_i^q)$)**.** *For any preemptable flow $f_i \in \mathcal{P}$, an upper bound on the maximum preemption overhead incurred by frame f_i^q (with $q \geq 1$) arriving at time a_i^q*

within the busy-period Δt, is given by Equation 4.20.

$$\mathrm{PO}_i^{\mathcal{P}}(\Delta t, q, a_i^q) = \underbrace{\frac{24\,bytes}{rTX}}_{(a)} \times \min\left\{ \underbrace{\left(\sum_{j \in \{\mathcal{F} \mid \mathrm{PC}_j < \mathrm{PC}_i\}} \eta_j^+(\Delta t) \right)}_{(b)}, \underbrace{N_i(q, a_i^q, \Delta t)}_{(c)} \right\} \tag{4.20}$$

Proof. The maximum preemption overhead is reached in one of two cases: (Case 1) All preemptable frames incur the maximum possible number of preemptions; or (Case 2) All incoming frames belonging to a higher preemption classes cause a preemption. For the first case, Equation 4.19 contained in term (c) provides an upper bound. For the situation described in Case 2, a computation of the maximum number of arrivals of frames in a higher preemption class is captured by term (b). Therefore, the actual number of preemptions cannot be greater than the minimum of these two terms. Now, since each preemption operation generates an overhead equal to term (a), the theorem follows. □

4.2.2.5 Worst-case queuing delay

So far, we have discussed the individual components of the worst-case queuing delay. For each frame f_i^q arriving at time a_i^q within the busy period, we obtain an upper bound $Q_i(q, a_i^q)$ for this factor by summing all these terms, as formally stated in Equation 4.21.

$$Q_i(q, a_i^q) = \begin{cases} \mathrm{LPB}_i^{\mathcal{E}} + \mathrm{SPB}_i^{\mathcal{E}}(q, a_i^q) + \mathrm{HPI}_i(Q_i(q, a_i^q)) & \text{if } f_i \in \mathcal{E}; \\ \mathrm{LPB}_i^{\mathcal{TP}} + \mathrm{SPB}_i^{\mathcal{TP}}(q, a_i^q) + \mathrm{HPI}_i(Q_i(q, a_i^q)) + \mathrm{PO}_i^{\mathcal{P}}(Q_i(q, a_i^q)) & \text{if } f_i \in \mathcal{TP}; \\ \mathrm{LPB}_i^{\mathcal{BP}} + \mathrm{SPB}_i^{\mathcal{BP}}(q, a_i^q) + \mathrm{HPI}_i(Q_i(q, a_i^q)) + \mathrm{PO}_i^{\mathcal{P}}(Q_i(q, a_i^q)) & \text{if } f_i \in \mathcal{BP} \end{cases} \tag{4.21}$$

Note that Equation 4.21 defines an iterative fixed-point process, since $Q_i(q, a_i^q)$ appears on both sides of the equation. Therefore, a valid solution for each frame f_i^q is obtained by starting the fixed point algorithm with the base value C_i^+. The algorithm stops as soon as two consecutive values of $Q_i(q, a_i^q)$ are identical or the relative deadline associated with the flow f_i is exceeded. In the latter case, the timing requirement of f_i is violated and there is no valid solution.

4.2.2.6 Worst-case traversal time

For each frame f_i^q arriving at time a_i^q, it follows from the discussions in the previous sections that its worst-case traversal time ($\mathrm{WCTT}_i(q,a_i^q)$) is given by the combination of the Equations 4.1 and 4.4, where $Q_i^+(q)$ (in Equation 4.1) and $\delta_i^-(q)$ (in Equation 4.4) are replaced by $Q_i(q,a_i^q)$ and a_i^q, respectively. Formally, $\mathrm{WCTT}_i(q,a_i^q)$ is given in Equation 4.22.

$$\mathrm{WCTT}_i(q,a_i^q) = \begin{cases} Q_i(q,a_i^q) + C_i^+ - a_i^q & \text{if } f_i \in \mathcal{E}; \\ Q_i(q,a_i^q) + \dfrac{84bytes}{rTX} - a_i^q & \text{if } f_i \in \mathcal{TP}; \\ Q_i(q,a_i^q) + \dfrac{84bytes}{rTX} - a_i^q & \text{if } f_i \in \mathcal{BP} \end{cases} \tag{4.22}$$

Finally, the worst-case traversal time for the flow f_i within the level-i busy period, denoted by WCTT_i^+, is obtained by computing the maximum of all $\mathrm{WCTT}_i(q,a_i^q)$, where $1 \le q \le q_i^+$ and q_i^+ is the last frame of f_i in the level-i busy period. Formally, this is expressed in Equation 4.23.

$$\mathrm{WCTT}_i^+ = \max_{1 \le q \le q_i^+} \left\{ \mathrm{WCTT}_i(q,a_i^q) \right\} \tag{4.23}$$

This equation completes the computation of WCTT_i^+ for the flow f_i within a switch node (i.e., the local analysis). Therefore, the overall traversal time for flow f_i (i.e., the global analysis) is obtained by adding these values at all individual switch nodes along its transmission path. Here, the level-i busy period is a time interval $[\sigma,\mu]$ in which only frames belonging to $hep(i)$ (except for the first frame generated by the flow $f_j \in lp(i)$ with the longest non-preemptable fragment) are transmitted throughout $[\sigma,\mu]$, but no frame belonging to $hep(i)$ is transmitted in $[\sigma-\varepsilon,\sigma)$ or $(\mu,\mu+\varepsilon]$ for any arbitrary small $\varepsilon > 0$.

To perform the analysis, we considered the scenario that results in the longest level-i busy period for each flow f_i was considered, i.e., the scenario in which either (1) a frame f_j^k in $\mathrm{PC}_i < \mathrm{PC}_j$ with the longest non-preemptable fragment; or (2) a frame f_j^k in $\mathrm{PC}_i = \mathrm{PC}_j$ with the largest size and a lower priority than f_i, was released just before f_i; all flows in $hep(i)$ release a frame at the same time as f_i and finally all future frames are released as soon as legally permissible.

4.3 Configuration

As with most, if not all, real-time and/or time-critical preemptive systems, an appropriate *priority-to-flow assignment* policy plays a central role in the resulting performance of both 1-level and multi-level preemption schemes to avoid over-provisioning and/or suboptimal use of hardware resources. In this scope, the so-called *Audsley's Optimal Priority Assignment algorithm* (AOPA) has become the reference in many real-time systems, provided there are no priority inversions (Park and Shin, 2019; Davis et al., 2007). Here, "optimality" refers to the ability of this algorithm to provide a priority-to-flow assignment that allows all flows to meet their timing requirements when such a scheme exists. Davis et al. (2016) noted that AOPA is not applicable in fixed priority schemes with differed preemptions and/or preemption thresholds, and this is the case with preemptive TSN. Therefore, the so-called *Deadline Monotonic Priority Ordering* (DMPO) (Davis et al., 2016) is often used in practice and is known to dominate most other priority assignment heuristics in terms of schedulability (Lee et al., 2021). Nevertheless, DMPO is also not suitable for priority assignment in preemptive TSN, as it provides a fully ordered priority list for the flowset. Ethernet only supports up to eight priority levels. With this limitation, a fully ordered priority list leads to another bin packing problem, which is known to be strongly NP-Complete (Lodi et al., 2002). An efficient priority assignment scheme should not only provide the best possible priority ordering for flows but also assign multiple flows to the same priority levels in the best possible way.

Note that *multi-level preemption scheme* brings a whole new dimension of configuration synthesis in addition to the challenge of priority assignment. In a 1-level preemption scheme, the configuration decisions are in fact quite simple and straightforward. They are limited to deciding whether each flow belongs to the express class or the preemptable class. The picture darkens considerably when multi-level preemption is adopted. In this case, the system designer must answer two important questions:

1. How should the number of preemption levels to be activated be determined?
2. How should the mapping between flow and preemption class be made?

In response to these concerns, it is worth noting that Ethernet can support at most a seven-level preemption scheme[2]. As reported in Chapter 3, each additional preemption level adds significant hardware overheads that can increase the switch manufacturing

[2]Ethernet has eight priority levels and thus at most eight preemption classes. Assuming that a flow in one preemption class can preempt any other flow in another preemption class with a lower priority, it follows that Ethernet can support at most a seven-level preemption scheme.

cost. To ensure optimal use of hardware resources, the system designer must ensure that only the required number of preemption levels are supported for transmitting flows over the network. Figures 4.5, 4.6, and 4.7 illustrate the importance of the preemption level synthesis problem.

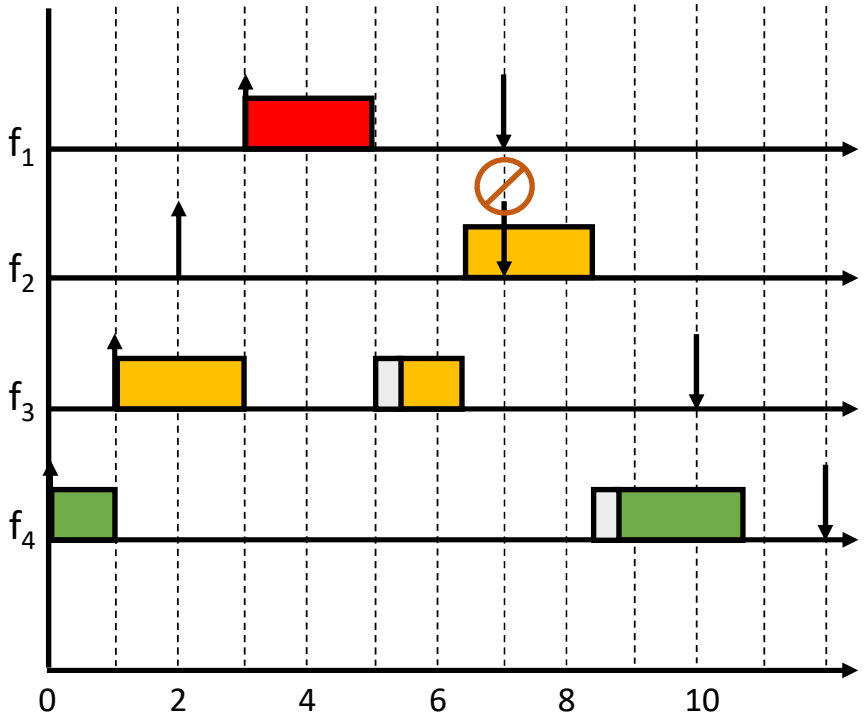


Figure 4.5: Flow f_2 misses its deadline with a 2-level preemption scheme.

In these figures, we consider the same four flows (f_1, f_2, f_3, f_4). We assume four possible preemption classes: (1) the *highest preemption class*, in which frames are represented by "red boxes"; (2) the *middle-high preemption class*, in which frames are represented by "yellow boxes"; (3) the *middle-low preemption class*, where frames are represented by "black boxes"; and finally (4) the *lowest preemption class*, where frames are represented by "green boxes". For this setting, there is no configuration under either a non-preemptive or a 1-level preemption scheme that allows all flows to meet their timing constraints. Assuming the multi-level preemption scheme, we analyze three configuration scenarios.

In Figure 4.5, flows are divided into three preemption classes and f_2 misses its deadline because it is unable to preempt f_3 – the two frames belong to the same preemption class. In Figure 4.6, the situation improves for f_2 and it meets its deadline, but at the cost of an additional preemption class. In Figure 4.7, an appropriate configuration (again with three preemption classes) is used and all frames meet their deadlines. From these obser-

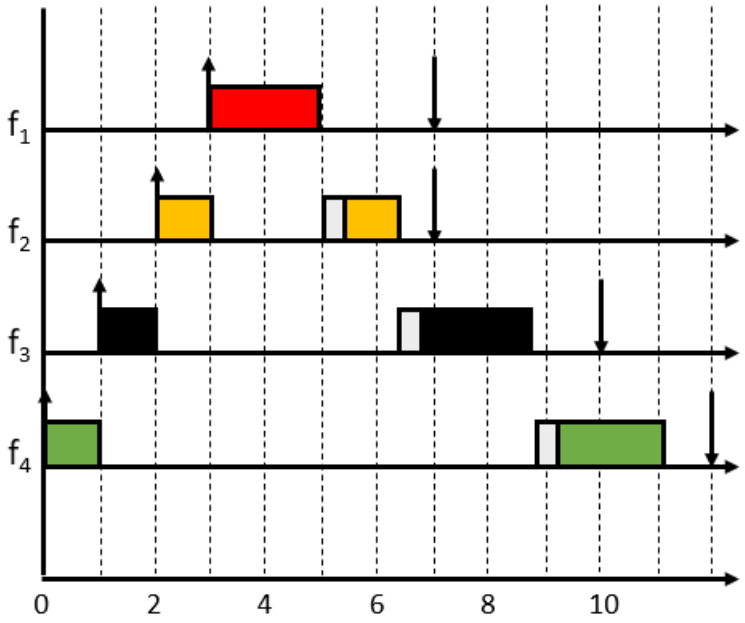


Figure 4.6: All flows meet their deadlines with a 3-level preemption scheme.

vations, it follows that the performance of the multi-level preemption system is highly dependent on the configuration chosen.

In this section, we advance the state of the art by addressing the above configuration issues. Given a set of flows and a TSN network, we first provide an offline priority assignment scheme for the flow set. Then, we provide an offline framework for determining the appropriate number of preemption levels on the one hand and the flow- to-preemption-class assignment on the other. Taken together, the proposed scheme has two goals: (1) to ensure that all flows meet their deadlines, and (2) to ensure that hardware resources are used efficiently.

4.3.1 Definitions

In addition to the system model in Section 4.1, we introduce the following definitions for any flow set $\mathcal{F}$ and network $\mathcal{G}$.

Definition 1 (Valid configuration)**.** *Any configuration will be stamped as "valid" if, upon the flow-to-preemption-class assignment, it conforms to Rules R_1, R_2 and R_3.*

Definition 2 (Solution)**.** *Any valid configuration will be considered a "solution" if all flows $f_i \in \mathcal{F}$ meet their deadlines.*

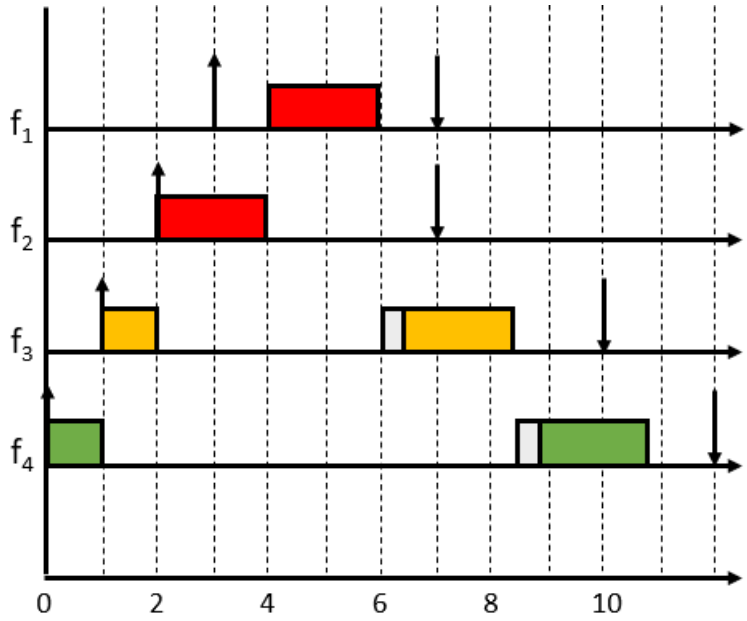


Figure 4.7: All flows meet their deadlines with a 2-level preemption scheme.

Rule R_2 from Section 4.1.3 implies that $\mathcal{C}_x^m$ (with $x \geq 1$) is always sorted in ascending order, i.e. $c_{x,\sigma}^m \leq c_{x,\ell}^m$ for all $0 \leq \sigma \leq \ell \leq p-1$. Rule R_3 implies that every integer from 0 to m must occur *at least once* in every valid configuration.

> Consequently, the task of generating all valid configurations $\mathcal{C}^m$ can be mapped to the problem of generating all ordered multisets (Blizard et al., 1989) of cardinality p with elements from the set $\{0,\ldots,m\}$, where p is the number of unique flow priorities in $\mathcal{F}$.

4.3.2 Proposed framework

In this section, we first provide a priority assignment scheme in Section 4.3.2.1, and then a preemption class assignment scheme in Section 4.3.2.2

4.3.2.1 Priority assignment

In this section, we present the priority assignment scheme for TSN flows based on the traditional *k-means clustering algorithm* (Hastie et al., 2009). This is a popular unsupervised Machine Learning (ML) algorithm that is scalable and robust to noise and outliers in the data. Briefly, the algorithm works as follows. It partitions the elements of an unlabeled list into k distinct clusters (with $k \geq 1$). Specifically, k foci, called “centroids," are iteratively computed within the data space. Clusters are formed around the centroids by

assigning each element to the centroid closest to it. Finally, the position of each centroid is updated after each iteration to the midpoint of all data points assigned to it.

> In the context of this work, the "elements" to be assigned are flows and the k clusters are the priorities.

We recall that Ethernet supports eight (8) priority levels, so $k \leq 8$. The k-means algorithm performs clustering based on some selected characteristics of the elements, called "features". These features capture the domain-specific knowledge about the real-world objects/concepts that the items represent. In this framework, the features need to be defined and sometimes transformed into a format that the k-means algorithm can process. The process of defining and preparing features for ML algorithms is called "feature engineering". In the following section, we summarize the feature engineering for our TSN priority assignment problem.

▷ **Feature engineering.** For the priority assignment problem, five of the seven flow parameters defined in the system model (see Section 4.1.2) are considered for the clustering process since the other parameters still need to be determined (i.e., priority and preemption class). In other words, the tuple $\langle src_i, dst_i, T_i, D_i, S_i \rangle$ is considered for each flow $f_i \in \mathcal{F}$. On the other hand, src_i and dst_i are used to determine the path length PL_i of the flow f_i, i.e., the number of links that f_i traverses from src_i to dst_i. Therefore, the number of features used for the clustering problem can be reduced to the tuple $\langle \mathrm{PL}_i, T_i, D_i, S_i \rangle$.

Note that these selected features are not on the same unit scale. In fact, the flow periods and deadlines can range from a few microseconds to hundreds of thousands of microseconds. At the other end of the table, the flow sizes can only take values between 64 bytes and 1500 bytes (i.e., the minimum and maximum valid Ethernet frame sizes, respectively). Finally, the range of PL_i largely depends on the network topology. In such a scenario, features with large values could dominate those with lower values, negatively affecting the actual performance of the k-means algorithm. To ensure that this is not the case, and to ensure that each feature has an impact on the learning process, a process called "normalization" is usually performed on the selected feature set (Hastie et al., 2009). In this work, this normalization process is performed for each of the features as follows. With respect to the flow length, all values are changed to obtain features in the interval $[-1, 0]$. For this purpose, we divide all flow lengths by the negative value of the longest flow path. Similarly, both T_i and D_i are normalized by dividing all values by the largest values of T_i and D_i, respectively, to obtain features in the interval $[0, 1]$. Finally, the values of S_i are normalized by the maximum frame size in the flowset to obtain fea-

tures in the interval $[0,1]$. The reason for normalizing the flow paths with the negative value of the longest flow path is that flows with shorter paths have higher feature values. This is explained in more detail in the following section. Note that with the normalization process, flows with longer paths, smaller periods, deadlines, and sizes are assigned smaller feature values.

▷ **Clustering and priority assignment.** Here we describe our strategy for assigning priorities to flows by partitioning the n flows in $\mathcal{F}$ into k clusters (with $1 \leq k \leq 8$) using the k-means algorithm. The overall goal is to assign priorities to flows in such a way that as many flows as possible can be scheduled, i.e., meet their deadlines. Figure 4.8 shows the assumed flowchart.

From the figure, it is clear that the process begins with the feature engineering process described above. After this step, two important questions need to be clarified.

> Q_1: How does one determine the value of k? In other words, how do you determine the number of clusters (priorities) into which to divide the flows?
>
> Q_2: How does one determine the relative order between clusters during the priority assignment process?

About Q_1. Since we do not know which value of k gives the maximum number of schedulable flows, we initialize k to 1 and iterate through all possible values ($1 \leq k \leq 8$).

About Q_2. At each iteration, we perform k-means clustering with the normalized features and obtain the centroid of each of the k clusters. Each centroid is a vector of the mean values of the features of the elements in the cluster. We then compute the mean of each centroid. Recall that because of the normalization process, flows with longer paths, smaller periods, deadlines, and sizes are assigned smaller feature values. Thus, the smaller the final value of the centroid of a given cluster, the longer the path and, in general, the smaller the period; deadline, and size of each of its members. In practice, flows with longer paths are at greater risk of not meeting their deadlines because of delays that may occur on the links they traverse. In addition, flows with shorter periods and deadlines are typically given higher priorities in fixed priority scheduling theory because they are at greater risk of missing their deadlines (think about Rate Monotonic and Deadline Monotonic policies). Finally, the sizes of non-time-critical flows are usually larger than the time-critical ones (see Lo Bello et al. (2020b); Ojewale et al. (2021) for examples inspired by real-world automotive use-cases).

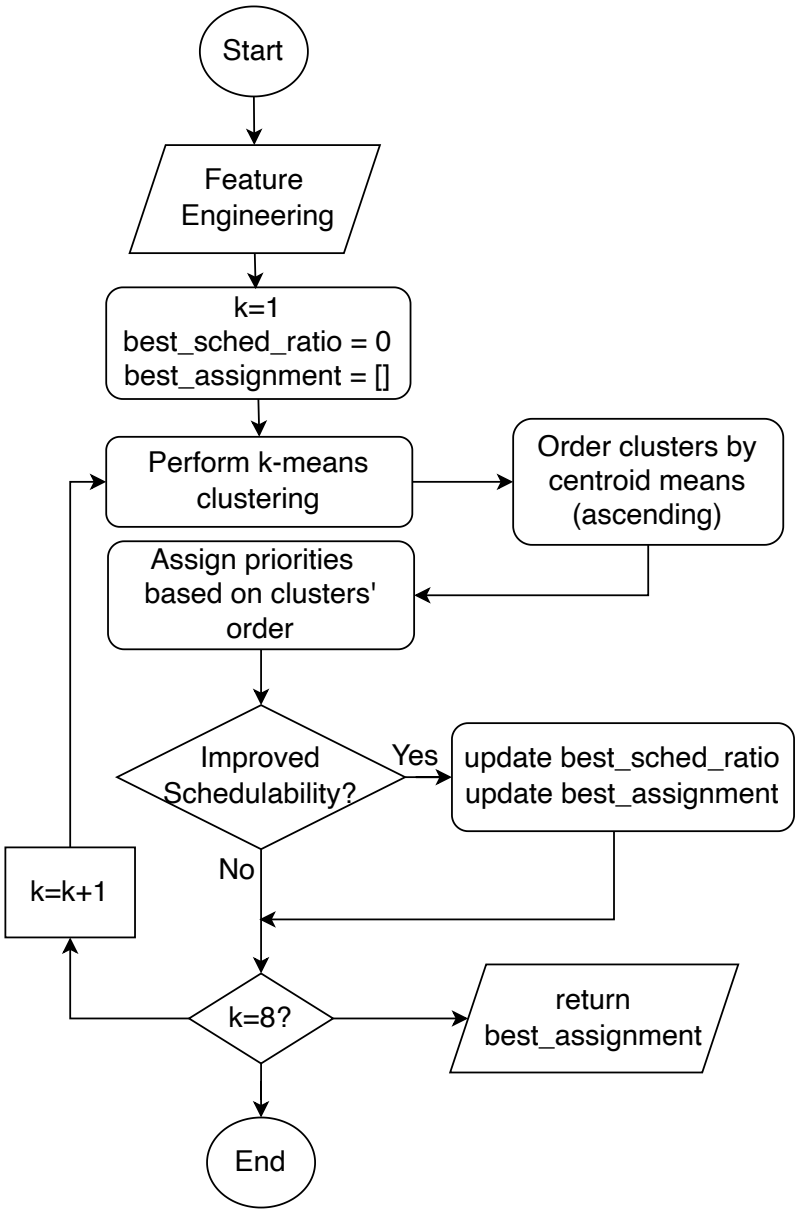


Figure 4.8: Flowchart: k-means priority assignment algorithm.

Based on the above observations, we chose to assign priorities to the clusters in ascending order of their centroid means. More specifically, *the lower the centroid mean, the higher the priority*.

In this work, we assume that all features have the same importance in the priority assignment process. Of course, other approaches to feature engineering can be used where specific properties of the network are exploited. For example, path length can be given a higher weight in a network with line topology. We also note that the schedulability tests

in the priority assignment process (see Figure 4.8) necessarily depend on the assignment of the preemption level, which is not defined at this point. To resolve this dependency, schedulability is evaluated assuming a fully preemptive scheme, i.e., any flow can preempt any other flow with a lower priority.

4.3.2.2 Preemption class assignment

▷ **Synopsis.** Our solution for preemption class assignment builds on the priority assignment scheme. It starts with a non-preemptive scheme and uses a guided exhaustive search approach described as follows. At each step, we introduce an additional preemption level and test all possible configurations of flow-to-preemption-class to verify that all timing requirements are met. Our algorithm terminates once a solution is found. Otherwise, another preemption level is added to the flow transmission scheme and a new test is performed. Note that we are only interested in determining the schedulability of a flow set and not whether there are multiple solutions for a given flow set. This process is iterated until a valid configuration is found for a given preemption level or the maximum number of preemption levels (i.e., 7) is exceeded. In the latter case, no valid configuration could be found and the flow set is classified as "unschedulable". In the following, we provide the step-by-step details of our proposed solution, which consists of two algorithms, namely (1) the *flow-to-preemption-class assignment*; and (2) the *valid configuration search*.

▷ **Flow-to-preemption-class assignment.** Algorithm 1 shows the pseudocode for this step. Two arguments are required as inputs, namely: (1) the network topology G; and (2) the flow set $\mathcal{F}$. In the description, the notation "P.Len" refers to the length of the list P and "$[y] * \eta$" refers to a list of length η filled up with "y", e.g., $[0] * 3 = [0,0,0]$. The algorithm proceeds as follows.

First, the set of all unique flow priorities in $\mathcal{F}$ is stored in the variable P (see line 1). Then, a systematic search through all preemption levels m (with $1 \leq m \leq \text{P.Len} - 1$) is performed for a solution (see lines 2 to 18). At each preemption level m, after the initialization phase (see lines 3 to 8), a recursive function VALID_CONFIGS() is called (see line 9). This function returns the set of all valid configurations for the preemption level m, i.e. $\mathcal{C}^m$. It then searches this set to find a solution (see lines 10 to 17). If a solution is found, Algorithm 1 terminates with that solution (see line 15), otherwise m is incremented and the process is repeated for the next preemption level $(m+1)$. If all iterations are exhausted and no solution is found, a NULL value is returned (see line 19), indicating that the flow set is "unschedulable". We note that for the schedulability test (see line 14) we use the worst-case traversal time (WCTT) analysis presented in Section 4.2.

Algorithm 1: ASSIGN_PREEMPTION_CLASS(G,$\mathcal{F}$)

Data: Network topology G; Flow set $\mathcal{F}$.
Result: A valid flow-to-preemption-class configuration.

```
1  P ← [Set of all unique flow priorities in F]
2  for m = 0 to P.Len − 1 do
3      C^m ← ∅
4      leftPart ← [ ]
5      rightPart ← [m] ∗ P.Len
6      for j = 0 to m do
7          rightPart[j] = j
8      end
9      C^m = VALID_CONFIGS(C^m, leftPart, rightPart)
10     foreach C^m_x ∈ C^m do
11         foreach f_i ∈ F do
12             PC_i = c^m_{x,P_i}
13         end
14         if SCHEDULABLE(F) then
15             return C^m_x
16         end
17     end
18 end
19 return NULL
```

▷ **Valid configuration search.** Algorithm 2 presents the pseudocode. The recursive function VALID_CONFIGS() computes the set of all valid configurations for a preemption level m. It takes 3 inputs: (1) an initialization of the set $\mathcal{C}^m$ (which is usually an empty set at the beginning); (2) a list leftPart (which is also usually empty at the beginning); and finally (3) a rightPart, which is a configuration initialization for the preemption level m. The notations in Algorithm 2 have the same meaning as those of Algorithm 1. In addition, the notation ListA + ListB implies a concatenation operation of two lists. In summary, VALID_CONFIGS solves the multiset generation problem described in Section 4.3.1. Since the algorithm is recursive, the first step is to test for the base case (see lines 2 to 5). This is the case when the elements in rightPart are either all the same or all unique. In this case, the algorithm terminates and returns a merged (and sorted) list of leftPart and rightPart. If the base condition is not satisfied, the recursive step (see lines 6 to 20) is executed.

▷ **Computational Complexity.** We recall that the problem of finding all valid configurations (Algorithm 2) has been mapped to the multiset problem, which is known to have exponential complexity. Stanley (2011) has already shown that the number of multisets

Algorithm 2: VALID_CONFIGS($\mathcal{C}^m$, leftPart, rightPart)

```
Data: Set of generated configurations C^m; a left partition; and an initial
      configuration for preemption level m
Result: Set of generated configurations C^m
1  rightPartSet = sorted(set(rightPart))
2  if rightPart.Len == rightPartSet.Len
     OR rightPartSet.Len == 1 then
3      C^m.add(sorted(leftPart + rightPart))
4      return C^m
5  end
6  diff = rightPart.Len − rightPartSet.Len
7  diff + = 1
8  currentLeft = rightPartSet.getFirstElem
9  rightPartSet.removeFirstElem
10 for x = 1 to diff do
11     tempLeft = [currentLeft] ∗ x
12     newLeftPart = leftPart + tempLeft
13     rightSize = rightPart.Len − x
14     newRightPart = list(rightPartSet)
15     if rightSize > newRightPart.Len then
16         lenDiff = rightSize − rightPartSet.Len
17         newRightPart = [rightPartSet.getFirstElem] ∗ lenDiff + list(rightPartSet)
18     end
19     C^m = VALID_CONFIGS(C^m, newLeftPart,
         newRightPart)
20 end
21 return C^m
```

of cardinality k to be taken from the set of m elements is given by:

$$\frac{(k+m-1)!}{k!(m-1)!}$$

We recall that each element of $\{1,2,\ldots,m\}$ must occur at least once in each multiset (see Section 4.1). This constraint reduces the number of elements whose order must be decided from k to $k-m$. Substituting k for $k-m$ gives the total number of valid configurations $|\mathcal{C}^m|$ for preemption m with k unique priorities in Equation 4.24.

$$|\mathcal{C}^m| = \frac{(k-1)!}{(k-m)!(m-1)!} \tag{4.24}$$

Algorithm 1 calls function VALID_CONFIGS() until a solution is found or the maximum number of preemption levels is exceeded. Consequently, the maximum number of valid

configurations to test is given by Equation 4.25.

$$\sum_{m=1}^{P.Len-1} |\mathcal{C}^m| \tag{4.25}$$

Chapter summary

In this chapter, we have provided formal and rigorous timing guarantees for each flow under a multi-level preemption scheme using a CPA-based approach and addressed the synthesis problem for multi-level frame preemption in TSN. Specifically, we first introduced the system model and assumptions and provided a background on the CPA model. Next, we identified the components of delay experienced by the flow and provided formal upper bounds for each of these components. Finally, we presented a configuration framework for multi-level preemption that addresses the unique configuration issues of the scheme. In particular, for a set of flows and the network topology, we have presented a framework for assigning priorities to the flows, determining which preemption level to enable, and assigning flows to preemption classes. Now that we have presented the formal timing analysis and configuration framework for determining priority assignment and a valid configuration for a given set of flows, we can demonstrate the applicability of the proposed framework and evaluate its performance in the next chapter.

Chapter 5

Evaluation

In the previous chapter, we proposed a WCTT analysis and configuration framework for the multi-level preemption scheme. In this chapter, we report the results of their evaluations from a quantitative perspective, using synthetic and real-world use cases. In particular, in Section 5.1, we evaluate the safety and tightness of the computed WCTT upper bounds and how they evolve under a multi-level preemption scheme compared to the non-preemptive and 1-level preemption schemes. We also evaluate the impact of each additional preemption level on the transmission of flows and assess the impact of the frame sizes in each preemption class. In Section 5.2, we evaluate the effectiveness of the priority and preemption class assignment schemes. For this purpose, our main evaluation metric is the schedulability ratio, i.e., the percentage (%) of flows that meet their timing requirements.

5.1 WCTT evaluation

In this section, we evaluate the safety and tightness of the proposed WCTT analysis. In Section 5.1.1, we consider a synthetic network modeled after a realistic automotive network, and in Section 5.1.2, we consider a Renault use case.

5.1.1 Report on a synthetic workload

▷ **Setup.** We consider a synthetic network that is adapted from realistic network topology from an automotive use case, consisting of ten End Points EPs (ECUs in this case) and seven full-duplex preemption-enabled TSN switches $SW_1, SW_2, \ldots, SW_7$ displayed as illustrated in Figure 5.1.

We have considered seven flows – $f_1, f_2, \ldots, f_7$ – where f_1 and f_2 are express; f_3 and f_4 are tpflows; and the remaining flows are bpflows. The name and make of the system

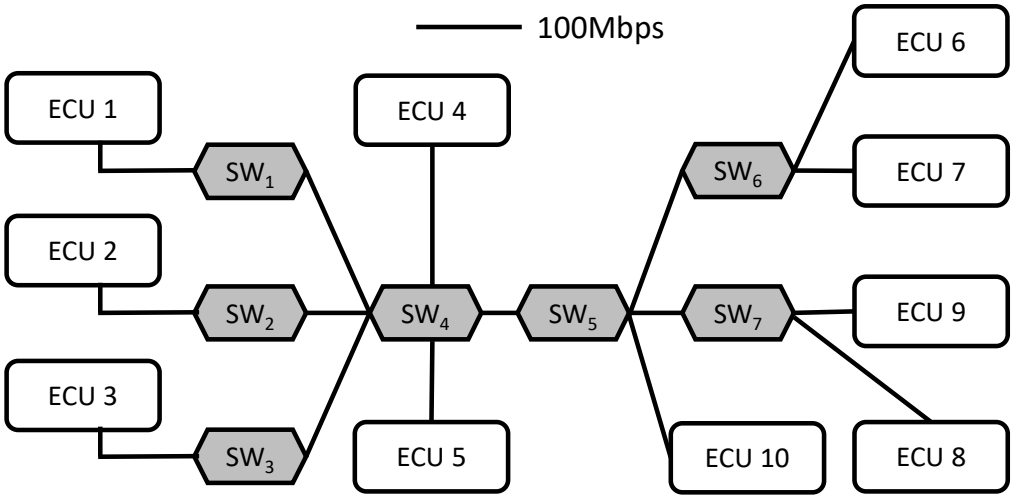


Figure 5.1: Network topology

are protected by a non-disclosure agreement. However, the specifications of the flows are given in Table 5.1 and are the same range as that presented by Alderisi et al. (2012). We recall that the flows are ordered by their priorities, i.e., the smaller the index of a flow, the higher its priority. Three batches of analyzes are performed along with their associated simulations using NeSTiNg (Falk et al., 2019), to evaluate their tightness:

(*a*) All flows are transmitted under the 1-level preemption scheme (i.e., only express flows can preempt other flows).

(*b*) All flows are transmitted under the 2-level preemption scheme (i.e., express frames can preempt all other frames; and tpflows can preempt bpflows).

(*c*) All flows are transmitted in a fully preemptive manner (i.e., any higher priority frame can preempt any lower priority frame).

▷ **Results and discussion.** Here we report the results we obtained in the experiments. We have tested the safety and tightness of the proposed analysis. We also report the behavior of the network with respect to "each additional preemption level" and the "maximum frame size in each preemption class".

▷◁ **On the tightness of the proposed analysis**. Table 5.2 compares the maximum measured end-to-end delays – mWCTT – with the analytical WCTT bounds.

Figures 5.2, 5.3, and 5.4 show the gaps between the measured end-to-end delays and the analytical WCTT values. We can see that all measured end-to-end delays (see the

Table 5.1: Flow properties

ID	Class	Source	Destination	Period (μ s)	Deadline (μ s)	Size (bytes)
1	express	EP 7	EP 3	5000	150	200
2	express	EP 6	EP 1	10000	200	250
3	tpflow	EP 2	EP 9	5000	500	300
4	tpflow	EP 3	EP 8	5000	500	400
5	bpflow	EP 4	EP 10	1000	-	1300
6	bpflow	EP 1	EP 8	1000	-	1300
7	bpflow	EP 9	EP 5	1000	-	1500

yellow box plots) fall below the corresponding WCTT (see the red dots) when using the 1-level preemption scheme (see Figure 5.2), the 2-level preemption scheme (see Figure 5.3), and finally the fully preemptive scheme (see Figure 5.4).

- Under the 1-level scheme, the observed gaps between the WCTT bounds and the mWCTT for the f_1 and f_2 express flows are 1.6% (WCTT : 120μs, mWCTT : 118μs) and 16.25% (WCTT : 144μs, mWCTT : 119.7μs), respectively. This pessimism stems from the fact that f_1 and f_2 share the same path as f_7. The gaps between WCTT and mWCTT for tpflows f_3 and f_4 are 30.92% (WCTT : 328μs, mWCTT : 226.56μs) and 43.05% (WCTT : 520μs, mWCTT : 296.16μs), respectively. This is due to the fact that tpflows use the same path as bpflows and can be blocked for long periods of time.

- Under the 2-level scheme, both the WCTT and the mWCTT remain the same for express flows. However, compared to the 1-level scheme, there is a significant improvement in the performance of tpflows. The maximum observed delay for f_3 and f_4 decreased by 21.17% (from 226.56μs to 178.58118μs) and 38.89% (from 296.16μs to 180.98μs), respectively. The WCTT and mWCTT gaps for f_3 and f_4 also decreased by 2.94% (WCTT : 184μs, mWCTT : 178.58μs) and 16.25% (WCTT : 244μs, mWCTT : 180.98μs), respectively.

Flows	**1-level preemption**		**2-level preemption**		**Fully preemptive**	
	WCTT (μs)	**mWCTT** (μs)	**WCTT** (μs)	**mWCTT** (μs)	**WCTT** (μs)	**mWCTT** (μs)
f_1	120.00	118.00	120.00	118.00	104.00	101.30
f_2	144.00	119.70	144.00	119.70	148.00	119.70
f_3	328.00	226.56	184.00	178.58	148.00	140.90
f_4	520.00	296.16	244.00	180.98	248.00	180.98
f_5	472.00	322.20	476.00	398.00	384.00	384.00
f_6	768.00	711.00	778.00	714.00	780.00	716.00
f_7	560.00	494.95	560.00	494.50	560.00	495.00

Table 5.2: Results from the synthetic workload: WCTT vs. mWCTT

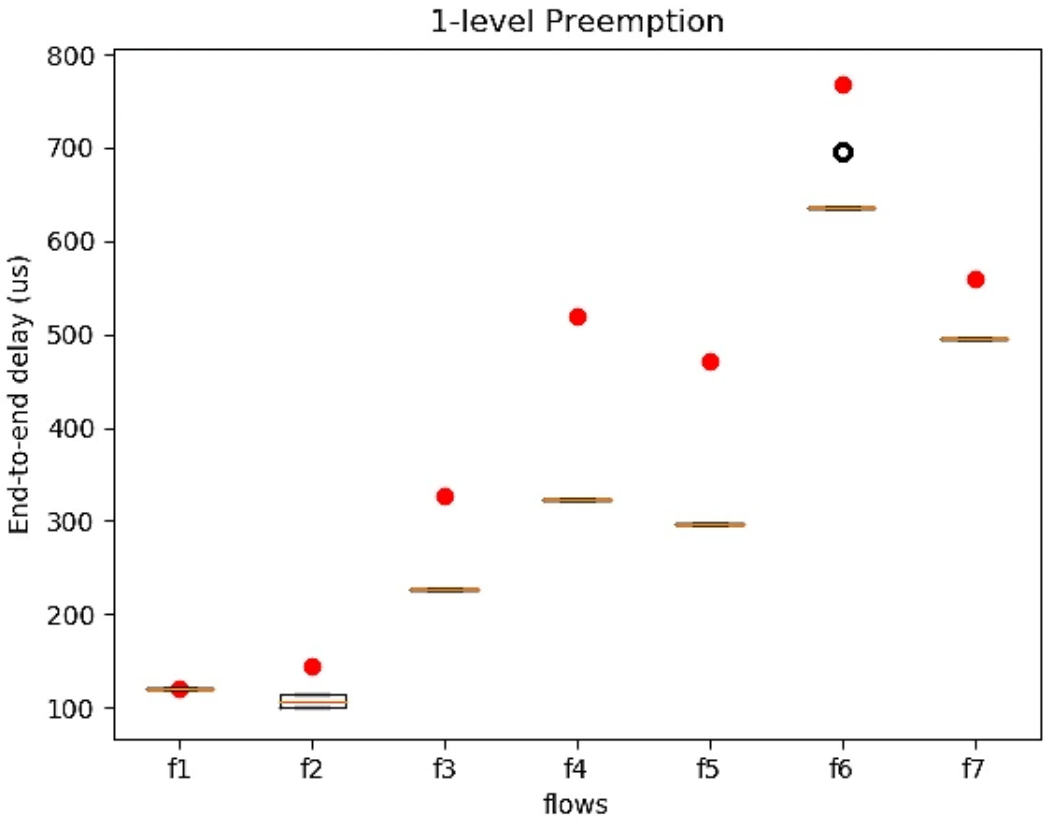


Figure 5.2: 1-level preemption scheme: Observed end-to-end delay from simulation. Red dots represent WCTT bounds, black are outliers.

- Under the fully preemptive scheme, there is further improvement in the performance of express flows and tpflows. In particular, the mWCTT values of f_1 and f_3 decreased by 14.1% (from 118μs to 101.3μs) and 21.09% (from 178.58μs to 140.9μs), respectively, compared to the 2-level scheme. The gaps between WCTT and mWCTT for flows f_1, f_2, f_3, and f_4 under the fully-preemptive scheme are 2.7% (WCTT : 104μs, mWCTT : 101.3μs), 17.77% (WCTT : 148μs, mWCTT : 119.7μs), 4.79% (WCTT : 148μs, mWCTT : 140.9μs), and 26.22% (WCTT : 248μs, mWCTT : 180.98μs), respectively. Of note is the slight degradation in the performance of f_2 and f_4 (1.67% and 1.10%, respectively). This is due to the overhead incurred by the additional preemption operations. From this observation, it follows that a tradeoff must be made between the number of preemption levels allowed for the flow transmissions and the additional overhead introduced by each new preemption level. Note that we can only define up to six intermediate levels of preemption since Ethernet offers a maximum of eight priority classes.

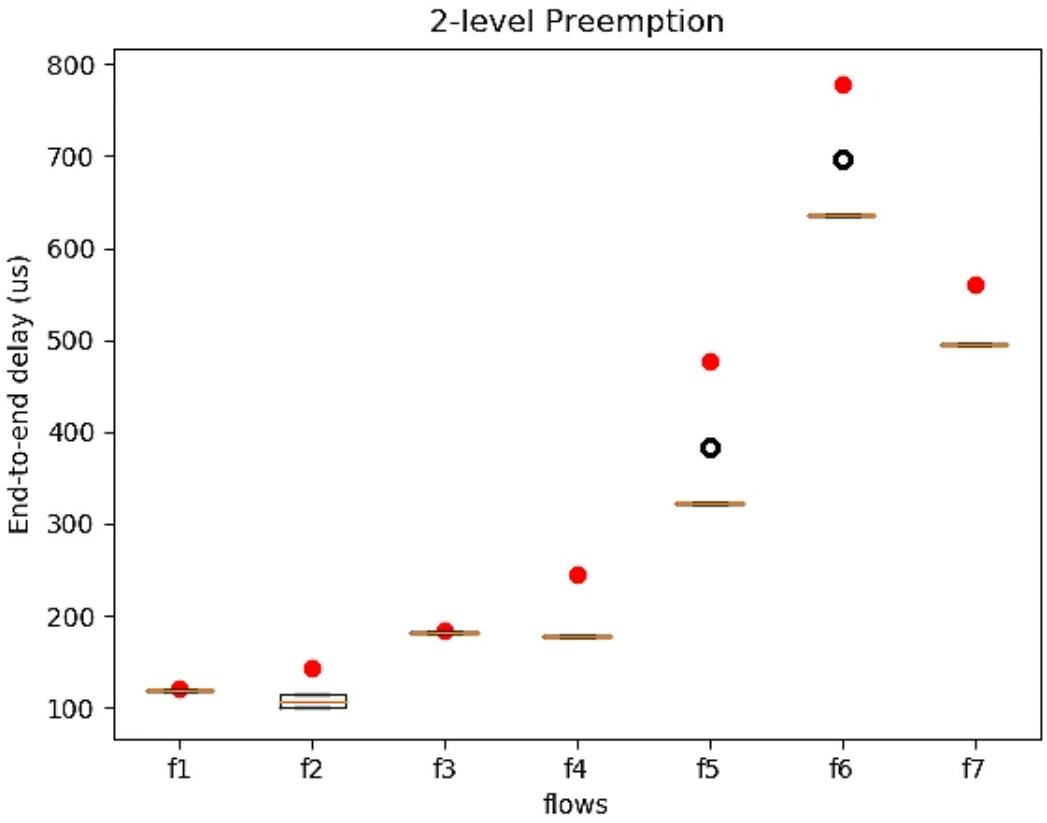


Figure 5.3: 2-level preemption scheme: Observed end-to-end delay from simulation. Red dots represent WCTT bounds, black are outliers.

⋈ **On the impact of each additional preemption level.** In Figure 5.5, we observe an improvement in the responsiveness of tpflows when an additional preemption level is added to the frame transmission scheme.

Specifically, this improvement reaches 43.9% (from 328μs to 184μs) and 53.07% (from 520μs to 244μs) for f_3 and f_4, respectively. The reasons for this trend can be explained as follows. The introduction of the additional preemption level protects f_3 and f_4 from possible long blocking periods associated with the transmission of f_6 and f_5 on their paths. We also note that the overhead introduced by this additional preemption level is negligible: the WCTT f_1 and f_2 remain the same for both the 1-level and 2-level preemption schemes. The same condition holds approximately for bplfows (i.e., flows f_5, f_6, and f_7). Here we find a cumulative performance degradation of 1.78%, which is negligible for the adopted use case (a degradation from 472μs to 476μs and 768μs to 778μs for f_5 and f_6, respectively). We also note that some mWCTT values for f_5 and f_6 (the black circles) fall outside the whiskers of the box plot. These outliers are the cases where an instance of a flow experiences an unusually high (or low) interference.

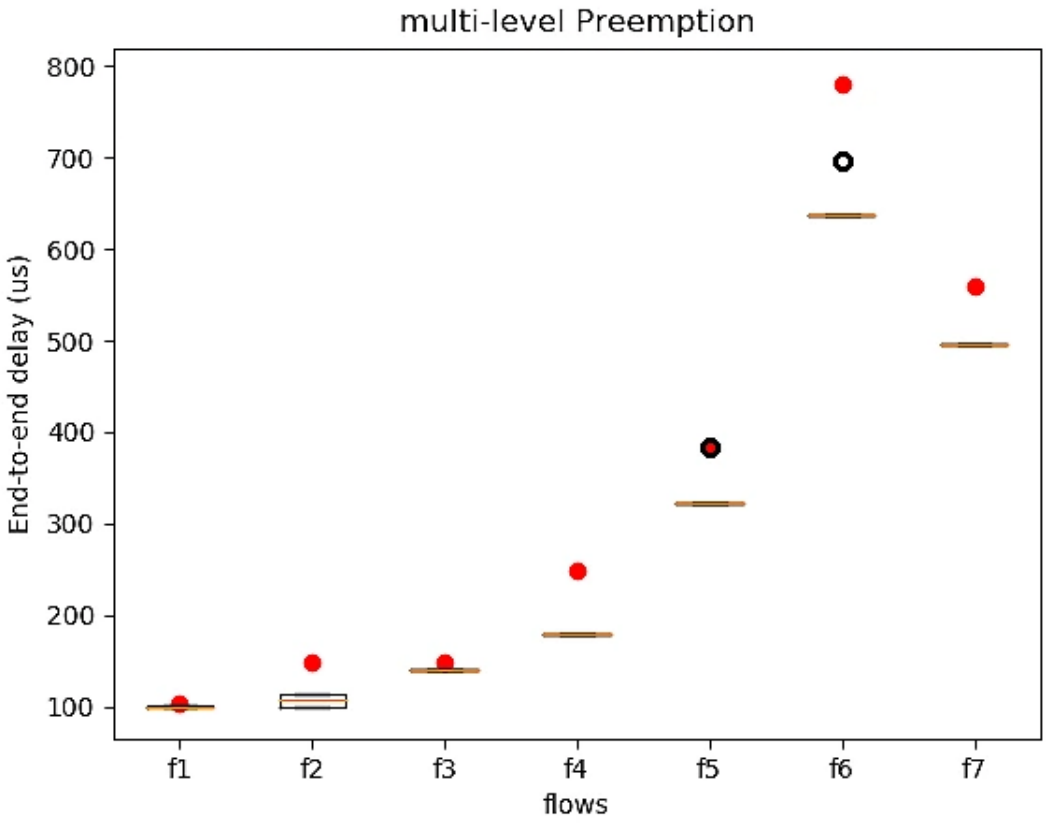


Figure 5.4: Fully-preemptive scheme: Observed end-to-end delay from simulation. Red dots represent WCTT bounds, black are outliers.

Still in Figure 5.5, we see an average improvement of 9.6% in the WCTT of tpflows when we switch from a 2-level preemption scheme to a fully preemptive approach. This suggests that the performance improvement is not linear, despite the benefits that each additional preemption level brings. On the downside, each preemption level introduces additional hardware implementation overheads that may prove to be non-negligible. This situation raises an open question: What is the optimal tradeoff in terms of the preemption-level scheme to be chosen for flow transmission, given that the performance gain achieved by enabling each additional preemption level is diminished by the associated hardware implementation cost?

> This question underscores the importance of the configuration framework presented in Chapter 4, which ensures that only the required preemption levels are activated.

▷◁ **On the impact of the frame sizes in each preemption class.** From the use-case setup, the maximum byte size of tpflows is reasonably small in comparison to that of the

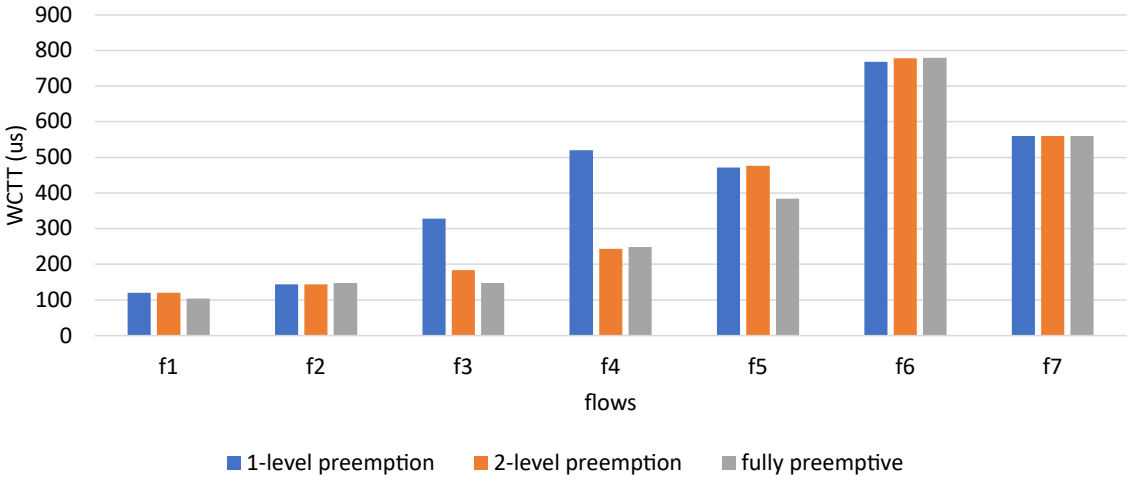


Figure 5.5: WCTT for each flow under 1-level preemption, 2-level preemption, and fully preemptive schemes.

bpflows. We vary this parameter from 400 to 1200 bytes in order to investigate its effect on the WCTT for each tpflow. The results are reported in Figures 5.6 and 5.7.

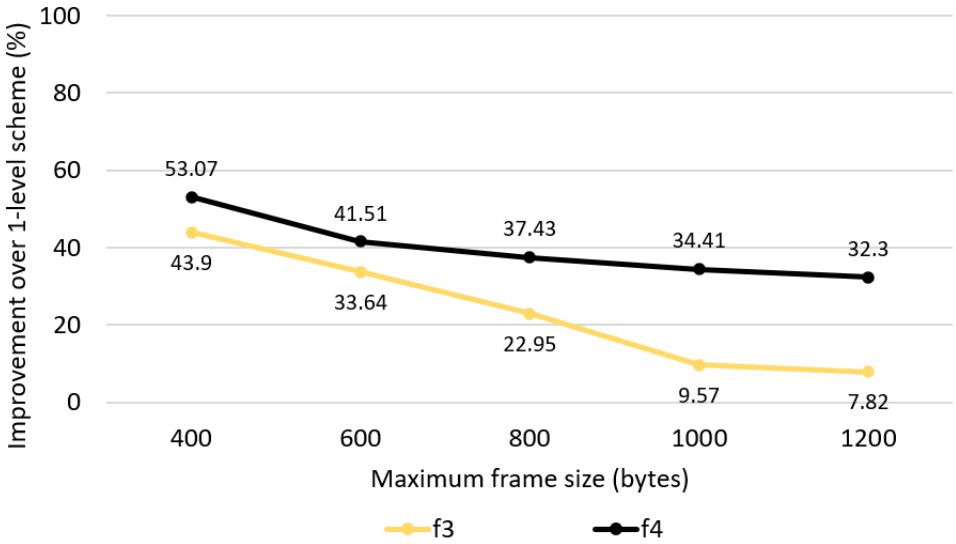


Figure 5.6: 2-level preemption scheme: Performance improvement w.r.t. maximum tpflow frame size.

In Figure 5.6, we see that the gain in WCTT obtained by the multi-level preemption scheme over the 1-level preemption approach decreases with an increase in the maximum frame size. Specifically, the improvement decreases from 43.9% (328μs to 184μs) to only 7.82% (614μs to 566μs) for f_3 and from 53.07% (520μs to 244μs) to 32.3% (972μs to 658μs) for f_4. This significant impact of frame size on the performance of f_3 can be

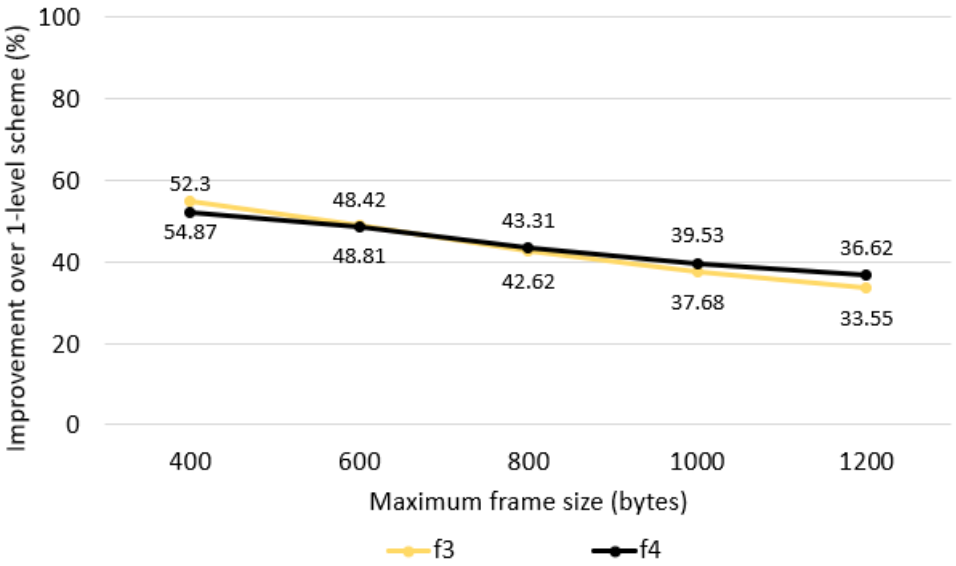


Figure 5.7: Fully-preemptive scheme: Performance improvement w.r.t. maximum tpflow frame size.

explained by the increasing blocking time it suffers as the highest priority tpflow. We recall that frames of the same preemption class are transmitted non-preemptively. Therefore, when assigning flows to preemption classes at design time, careful attention must be paid to the maximum possible frame size of each preemption class. It is worth noting that the degradation is less severe when a fully preemptive scheme is adopted, as shown in Figure 5.7. Here, the performance gain over the 1-level preemption scheme decreased from 54.87% (328μs to 148μs) to 33.55% (614μs to 408μs) for flow f_3 and from 52.30% (520μs to 248μs) to 36.62% (972μs to 616μs) for flow f_4.

5.1.2 Report on a Renault automotive network.

To further evaluate the safety of WCTT bounds, we consider another use-case scenario with a larger network and more flows, provided by Renault and borrowed from Migge et al. (2018). The assumed network topology is shown in Figure 5.8.

▷ **Setup.** The network includes 5 full-duplex Ethernet switches and 14 nodes: 4 cameras (CAMs), 4 displays (DSPs), 3 control units (ECUs), and 3 (functional) domain masters (DMs). The data transmission rate is 100Mbit/s on all links. Under this assumption, we consider a fully preemptive scheme, i.e., the preemption class of each flow is also its priority. The traffic specification consists of a total of 41 flows, as shown in Table 5.3.

▷ **Results and discussion.** Figure 5.9 presents the results obtained from the experiments.

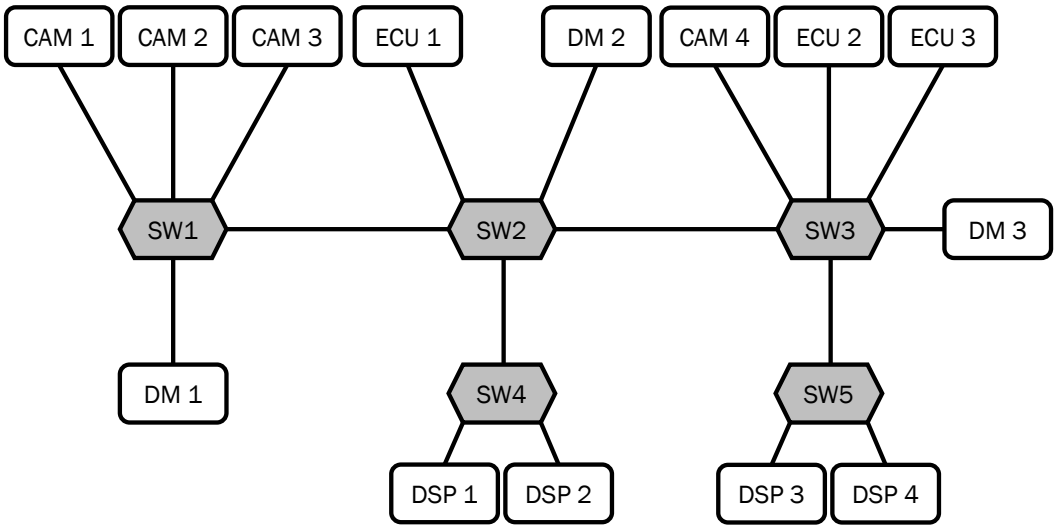


Figure 5.8: Network topology

Audio streams	– 8 streams – 128 and 256 byte frames – up to sub-10ms period and deadline – soft deadline constraints
Command and Control (C&C)	– 11 streams 256 to 1024 byte frames – up to sub-10ms periods and deadlines – deadline constraints (hard)
Video streams	– 2 ADAS + 6 Vision streams – up to 30*1446 byte frame each 16ms (60FPS) or each 33ms (30FPS) – 10ms (ADAS) or 30ms deadline (Vision) – hard and soft deadline constraints
Best-effort: file & data transfer, diagnostics	– 11 streams including TFTP traffic pattern – up to 0.2ms period – both throughput guarantees (up to 20Mbits per stream) and deadline constraints (soft)

Table 5.3: Prototype flow specification with the characteristics and performance requirements for each traffic class.

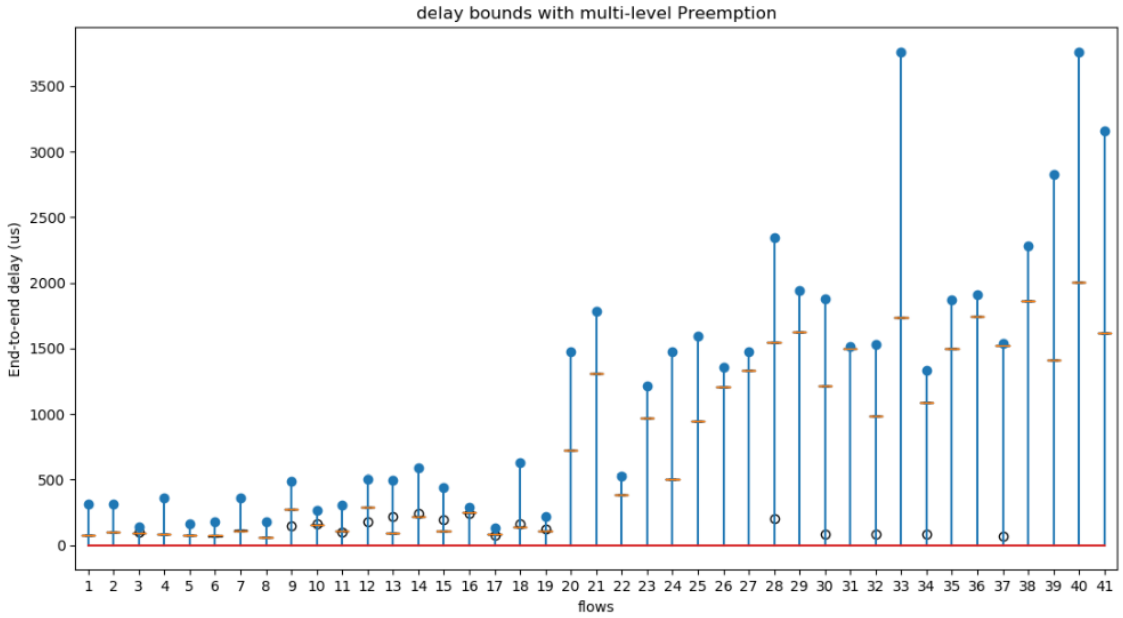


Figure 5.9: Observed end-to-end delays from simulation for the Renault use-case (in yellow box plots). The blue lines represent the WCTT bounds and the circles represent outlier values.

The numerical values are listed in Table 5.4. From Figure 5.9 it can be seen that the mWCTT of all flows are smaller than the computed WCTT bounds. This also gives further confidence on the safety of the analysis presented in this work. The mWCTT values of Audio flows – which are the flows with the highest priority - are on average 65.92% lower than the WCTT values. The mWCTT values for Command and Control Traffic – the highest priority traffic after Audio traffic – are on average 42.84% lower than the WCTT values. Finally, the mWCTT values for the lower priority flows (Video and Best Effort flows) are on average 36.53% lower than the WCTT values.

We found that the distance between the mWCTT values and the WCTT values is smallest for the lowest priority flows (Best Effort flows), and that the gap between the mWCTT and the WCTT values generally increases as the priority of flows increases. The reason for this is that lower-priority flows are more likely to be exposed to the maximum possible interference, as considered in the analysis.

mWCTT and WCTT for the Renault Automotive Use-Case								
ID	Src.	Dest.	Size (bytes)	Priority	period (μs)	Deadline (μs)	mWCTT (μs)	WCTT (μs)
1	DM2	DSP1	210	1	5000	5000	72.5	310
2	DM1	DSP1	200	1	5000	5000	95.8	310
3	DM1	DM3	189	0	5000	5000	97.5	136
4	DM2	DSP2	199	1	5000	5000	87.0	363
5	ECU1	DSP2	179	0	5000	5000	77.5	163
6	DM2	DSP2	159	0	5000	5000	75.8	177
7	DM2	DSP2	225	0	5000	5000	115.4	362
8	DM2	DSP1	138	0	5000	5000	59.5	177
9	ECU3	DSP1	240	2	10000	10000	304.0	490
10	ECU3	DM1	170	3	10000	10000	165.2	263
11	ECU3	DM3	239	3	10000	10000	112.6	204
12	ECU2	DSP2	200	2	10000	10000	322.0	499
13	ECU1	DSP2	234	2	10000	10000	252.0	492
14	ECU1	DSP2	214	3	10000	10000	247.0	589
15	ECU1	DSP1	210	2	10000	10000	214.0	442
16	ECU1	DM1	190	3	10000	10000	251.2	289
17	ECU3	DM3	210	2	10000	10000	90.2	133
18	ECU3	DSP1	242	3	10000	10000	167.5	630
19	ECU1	DM1	250	2	10000	10000	121.8	215
20	CAM4	DSP4	1446	5	10000	10000	723.2	1475
21	CAM1	DSP1	1446	5	10000	10000	1304.9	1783
22	CAM4	DSP4	1446	4	10000	10000	382.6	525
23	CAM1	DSP3	1446	4	10000	10000	965.8	1217
24	CAM2	DSP2	1446	4	10000	10000	505.9	1477
25	CAM1	DSP2	1446	4	10000	10000	941.1	1597
26	CAM4	DSP3	1446	5	10000	10000	1208.83	1355
27	CAM4	DSP4	1446	5	10000	10000	1329.6	1475
28	CAM1	DM3	1446	6	10000	10000	1529.7	2341
29	ECU2	DM1	1446	7	10000	10000	1621.1	1939
30	CAM1	DM2	1446	6	10000	10000	1500.6	1880
31	CAM2	DM1	1446	7	10000	10000	1499.8	1518
32	CAM2	DM2	1446	6	10000	10000	1220	1528
33	CAM2	DM3	1446	7	10000	10000	1737.5	3757
34	ECU2	DM2	1446	7	10000	10000	1076.2	1330
35	CAM3	DM3	1446	6	10000	10000	1494.9	1867
36	ECU3	DM1	1446	7	10000	10000	1742.4	1906
37	CAM2	DM1	1446	7	10000	10000	1509.7	1518
38	CAM4	DM1	1446	7	10000	10000	1863.7	2279
39	CAM2	DM2	1446	7	10000	10000	1410.7	2826
40	CAM2	DM3	1446	7	10000	10000	2005.6	3757
41	CAM3	DM3	1446	7	10000	10000	1616.2	3157

Table 5.4: Results: Renault use-case

5.2 Configuration framework evaluation

In this section, we evaluate the performance of the priority assignment algorithm and demonstrate the applicability of the preemption class assignment framework. In Section 5.2.1 we consider a synthetic network and a synthetic workload, and in Section 5.2.2 we consider two real-world use cases: the SAE automotive benchmark and the Orion Crew Exploration Vehicle (CEV).

5.2.1 Evaluation on a synthetic workload.

▷ **Setup.** We consider a quad-star topology consisting of six EPs and three TSN switches connected as shown in Figure 5.10. The link speeds are assumed to be constant and fixed

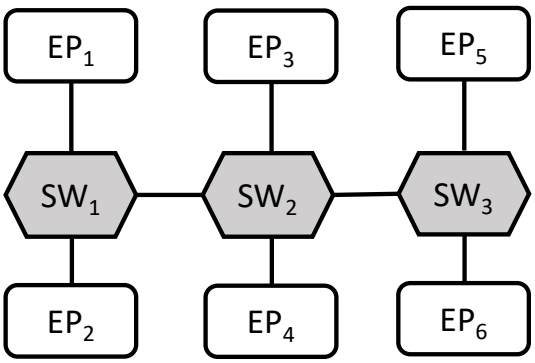


Figure 5.10: Synthetic network with quad-star topology.

at 100MBits/s. In each batch of experiments, we randomly generated 1000 flowsets of equal size, i.e., each flowset has exactly the same number of flows as the others. We varied the sizes between 100 and 250 flows per flowset. Each generated flow is characterized by a source, destination, period, deadline, and size. The sources and destinations are randomly selected among the EPs. The values for the periods and deadlines range from $500\mu s$ to $100000\mu s$. The values for the flow sizes range from 64 bytes to 1500 bytes. We then assign priorities to the flows using our proposed approach. The priority assignment framework was implemented in Python 3.6, and `scikit-learn` – a free machine learning software library for the Python programming language – was used for k-means clustering. We compare our solution to DMPO because it is known to outperform most other priority assignment schemes in the literature (Lee et al., 2021; Davis et al., 2016). In addition, DMPO has been specifically recommended for preemptive schemes with fixed priority and preemption thresholds (Davis et al., 2016), a model very similar to the one considered in this work. Since DMPO provides a fully ordered priority list for the flowset

and Ethernet only supports up to eight priority levels, we perform greedy bin packing of the ordered flows by assigning the same number of flows to each priority level. Since the exact number of priority levels that gives the best performance with DMPO is not known, we tried all possible numbers of priority levels k (with $1 \leq k \leq 8$) and report the best performance.

> Our main evaluation metric is the schedulability ratio, i.e., the percentage (%) of flows that meet their timing requirement under the priority assignment scheme.

To evaluate the schedulability of each scheme, we use the worst-case traversal time (WCTT) analysis presented in Chapter 4. For both our k-means approach and DMPO, we ran each of the experiments multiple times and then report the average observed performance on schedulability. In the first set of experiments, we assign priorities to the flows using DMPO and k-means, and evaluate the schedulability of the flows under the assumption of a fully preemptive scheme, i.e., a flow can preempt any other flow with a lower priority than itself. In the second set of experiments, we applied our preemption class assignment algorithm to determine the appropriate preemption level for each flowset.

▷ **Results and discussion.** Figure 5.11 shows the schedulability results under the k-means and DMPO schemes.

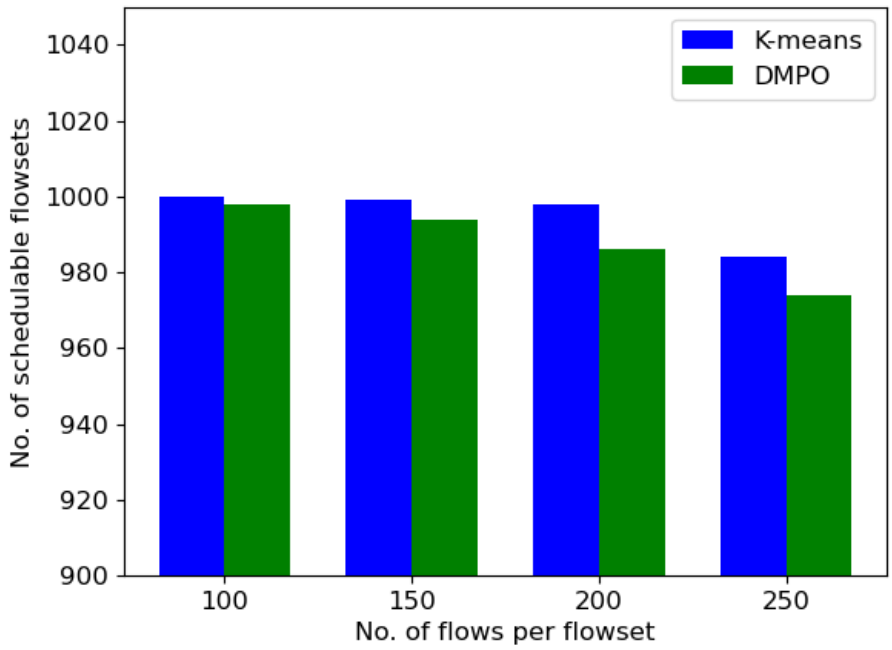


Figure 5.11: Schedulability: K-means vs. DMPO.

From the figure, it can be seen that k-means is able to schedule a higher number of flows than DMPO. Specifically, the average number of schedulable flowsets ranges from 999.5 to 981 for the k-means scheme and from 998 to 974 for DMPO. Figure 5.12 shows the average runtime per flowset for both k-means and DMPO.

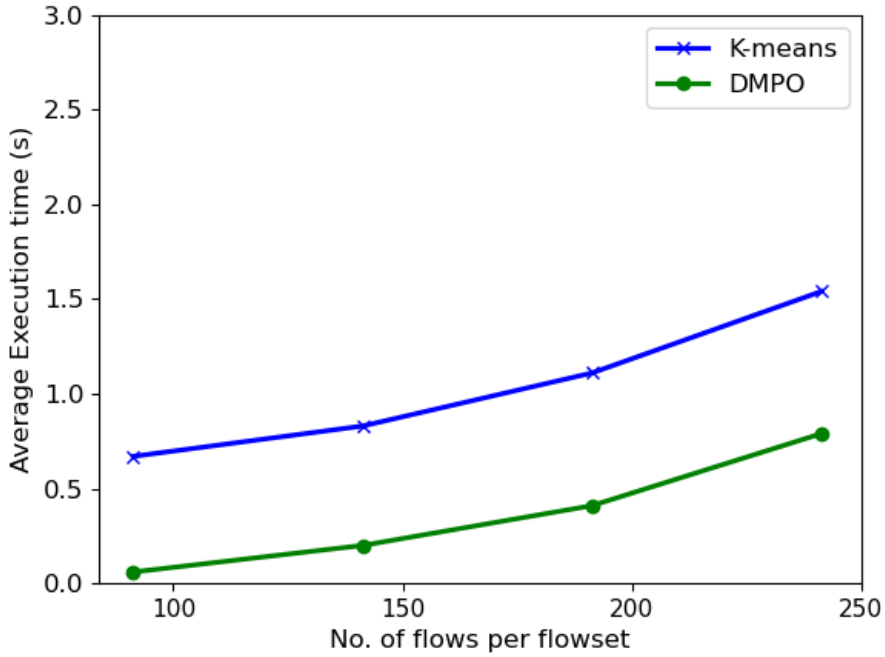


Figure 5.12: Runtime: K-means vs. DMPO.

The reported execution times were obtained from a commodity hardware (Intel(R) Core(TM) i7-6500U CPU @ 2.50GHz, 16GB memory). From Figure 5.12 it can be seen that DMPO is much faster than the k-means algorithm. This is due to its complexity.

> The core of DMPO is a sort with complexity $O(nlog(n))$, while k-means is known to have complexity $O(n^2)$ (Pakhira, 2014). However, we note that k-means is still quite fast for the priority assignment task with an average runtime of 1.6s compared to DMPO's 0.79s for flowsets of 250 flows each.

Figure 5.13 shows the schedulability results for the preemption level assignment (Algorithm 1). The figure shows that the average number of schedulable flowsets increases as the preemption level increases. Specifically, the schedulability ratio jumps from 45.5% under the non-preemptive scheme to 85.8% under the 1-level preemption scheme, and the trend continues, albeit non-linearly, with each successive preemption level to peak at 98.1% under a fully preemptive scheme. It follows from these observations that the limitations of the 1-level preemption scheme can result in a non-negligible number of flowsets (13% in this case) being unschedulable, despite the enormous improvements it brings over

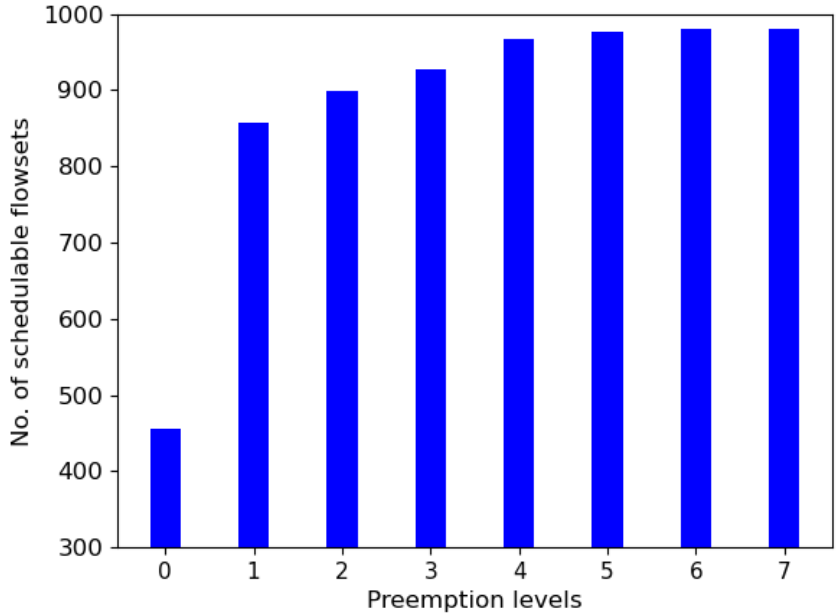


Figure 5.13: Schedulability results w.r.t. increasing preemption levels

the non-preemptive scheme. This problem is circumvented by the multi-level preemption scheme. Note that the improvement over the 4-level preemption scheme is not significant and/or provides limited benefit in this case. In addition, it is important to recall that each preemption level introduces additional hardware implementation overhead.

The configuration framework is useful to evaluate the trade-off between the preemption level scheme to be used and the performance gain from enabling each additional preemption level.

Figure 5.14 shows the average runtime per flowset for each preemption level configuration.

From the figure, computing the preemption configuration and evaluating the schedulability of a flowset with 250 flows under a non-preemptive scheme takes 1.6s on average. The execution time increases slowly as new preemption levels are added, reaching a peak of 6.21s on average per flowset under a fully preemptive scheme. This shows that despite the fact that the preemption level algorithm is a guided exhaustive search approach, its execution time is not prohibitive, making the proposed scheme applicable in real-world scenarios.

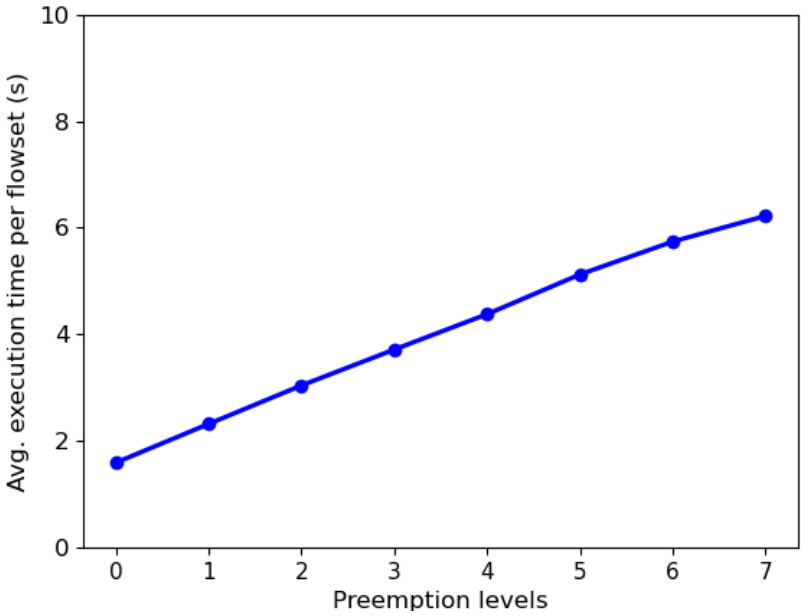


Figure 5.14: Average runtime w.r.t. increasing preemption levels

5.2.2 Evaluation on real-life use-cases

▷ **Setup.** SAE is the "SAE automotive communication benchmark" and Orion is the embedded communication network of the Orion Crew Exploration Vehicle (CEV). The network topologies for the SAE and Orion use cases are shown in Figures 5.15 and 5.16, respectively.

In the figures, EPs are represented by rectangles and TSN switches by hexagons. The flow parameters are given by Gavriluţ and Pop (2020)[1]. The link speeds are assumed to be constant and fixed at 100MBits/s. In the first set of experiments, we assign priorities to the flows using the two priority assignment schemes (k-means and DMPO) and evaluate the schedulability of the flows by assuming a fully preemptive scheme. In the second set of experiments, we applied our preemption class assignment algorithm to determine the preemption level where the number of schedulable flows is the highest.

▷ **Results and discussion.** Table 5.5 shows the schedulability performance of the two priority assignment schemes evaluated.

Name	EPs	SWs	No. of flows	K-means	DMPO
SAE	15	7	79	97.46 %	93.67%
Orion CEV	31	15	184	86.9%	80.4%

Table 5.5: Experimental results from real-life use-cases

[1]The files for all test cases are available at https://bit.ly/2kpLrKj.

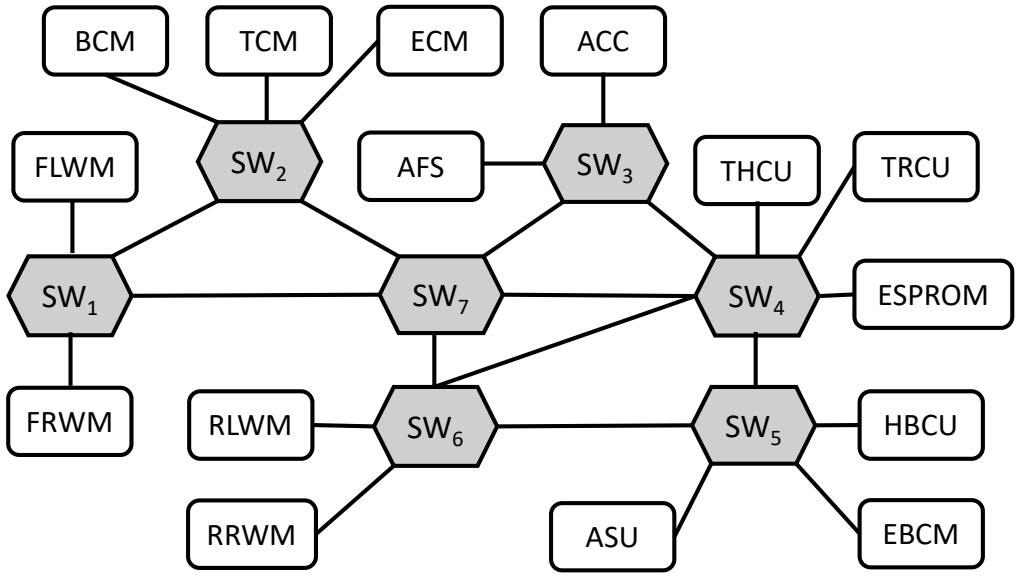


Figure 5.15: SAE automotive communication benchmark topology Gavriluţ and Pop (2020)

The table shows that the k-means priority assignment scheme again outperforms DMPO in both use cases. More specifically, k-means was able to schedule 97.46% and 86.9% compared to DMPO's 93.67% and 80.4% for the SAE and Orion networks, respectively. For the SAE benchmark network, the performance of the k-means algorithm in terms of schedulability matches that reported by Gavriluţ and Pop (2020), which used a traffic type assignment (TTA) approach to find a feasible schedule. As far as we know, this is the best performance reported in the literature. Figure 5.17 shows the behavior of the SAE and Orion setups as the number of preemption levels increases.

From the figure, we see that frame preemption significantly increases the number of schedulable flows for both SAE and Orion. We also note that the number of schedulable flows continues to increase in the Orion use case until it reaches a peak value of 86.9% for a 3-level preemption scheme. On the other hand, the schedulability of flows in the SAE use case peaks at a 1-level preemption.

Observations like these can help system designers to decide the best number of preemption levels to enable for a system at design time and ensure that only the needed preemption levels are implemented.

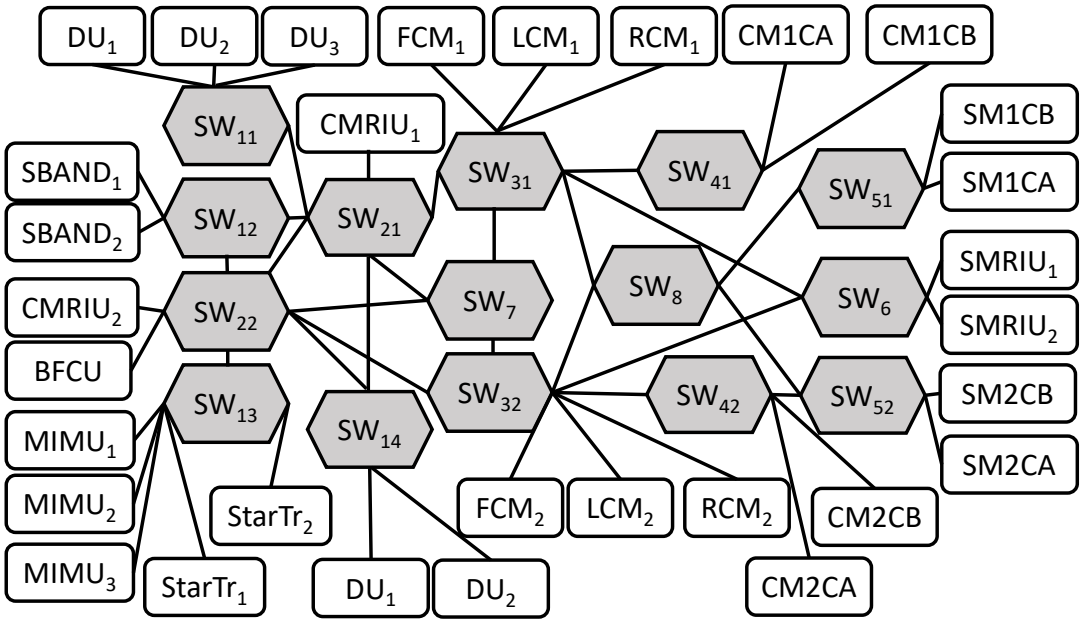


Figure 5.16: Orion CEV network topology (Zhao et al., 2017).

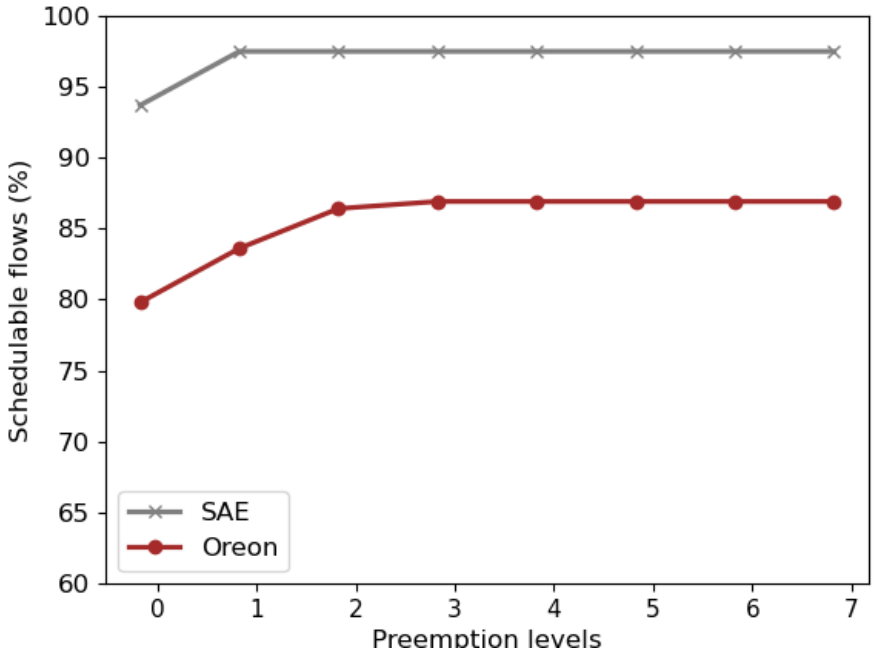


Figure 5.17: % of schedulable flows w.r.t. increasing preemption levels for SAE and Orion.

Chapter Summary

In this chapter, we experimentally evaluated the safety of the WCTT bounds and the performance of the proposed framework. Using a realistic automotive use case, we evaluated the performance improvements in terms of worst-case traversal time (WCTT) over the 1-level scheme. Our results show that the multi-level preemption scheme has an improvement of up to 53.07% in the WCTT guarantee for preemptable time-sensitive frames. We then demonstrated the safety of the analysis using another use case from Renault. We concluded that the maximum size of each of these flows must be carefully considered when assigning flows to preemption classes at design time. We also discussed the performance improvement achieved by each additional preemption level in the frame transmission scheme. We have shown that, on the one hand, the trend is not linear and, on the other hand, each additional preemption level introduces additional hardware implementation costs that may not be negligible. Our results show that the proposed upper bounds are safe and that the WCTT values of time-sensitive preemptable flows are much lower for a multi-level preemption scheme than for a 1-level preemption scheme.

> Our results show that the proposed priority-assignment scheme outperforms the DMPO scheme, which is known to dominate most other priority-assignment schemes in the literature. We also show that the proposed framework minimizes the number of preemption levels to ensure schedulability.

This is particularly important since each additional preemption level is associated with significant hardware overheads that can increase the cost of manufacturing switches. Furthermore, our proposed framework demonstrates acceptable scalability for practical use cases. In the next chapter, we consolidate the observed performance improvements by providing routing heuristics to further improve the responsiveness and reliability of frame transmission in TSN.

Chapter 6

On Smart Routing Schemes for Performance Improvement

In the previous chapters, we implicitly assumed that for each flow there is a predefined and/or known route from the source node to the destination node, and that the routing strategy does not affect the performance of the network. Nayak et al. (2018), Singh (2017), and Gavrilut et al. (2017), among others, have highlighted the importance of flow routing schemes for low latency, predictability, and reduced architectural cost in TSN, arguing that an inappropriate routing strategy may increase the number of transmission operations, leading to additional delays. In fact, an inappropriate routing scheme can increase the blocking time of flows in the network if too many flows attempt to traverse the same path at the same time. In this chapter, we focus on the routing problem of TSN flows. We believe that a strategy that minimizes the number of transmissions and the blocking time of each flow would help to avoid or mitigate these situations.

The TSN standard for path control and reservation (IEEE, 2016d) recommends the *Constrained Shortest Path First* (CSPF) routing scheme for transmission of time-sensitive flows (see page 71). It specifies that this scheme

> " *essentially performs shortest-path routing on the topology that only contains the links meeting the constraint(s).* "

From this quote, it is clear that CSPF is similar in its operation to the *Shortest Path Algorithm* (SPA). Consequently, this algorithm is also susceptible to congestion and longer blocking times for flows. To illustrate this claim, consider the network topology in Figure 6.1, in which six nodes (N_1 to N_6) and four switches (SW_1 to SW_4) are connected by full-duplex links. The nodes communicate through flow transmissions across the links and switches.

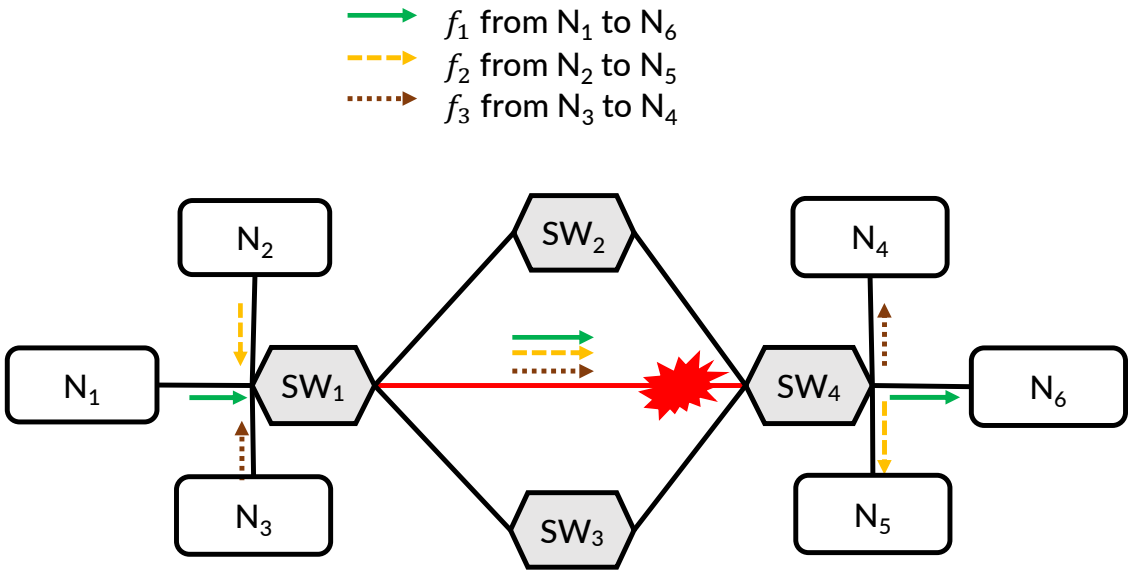


Figure 6.1: Congestion under CSPF routing policy.

In this example, we consider three flows – flow f_1 (green) is transmitted from N_1 to N_6; f_2 (yellow) from N_2 to N_5; and finally, flow f_3 (brown) is transmitted from N_3 to N_4. We assume that the CSPF routing strategy is applied and that all valid paths from each source to each destination node allow each flow to satisfy its end-to-end timing requirement. Then, all these flows are transmitted over their shortest paths, which include the "direct link" (in red) between SW_1 and SW_4, thereby increasing the eventual blocking time over this link for each flow. This fact makes this link a potential single point of failure for the network and can cause congestion despite the high connectivity. The same limitation applies to the so-called *Equal Cost Multi-Path* (ECMP) and the *weighted Equal Cost Multi-Path* (wt-ECMP) routing schemes (Singh, 2017). The basic idea of these two routing schemes is as follows. In ECMP, instead of computing a single shortest route, as is the case with SPA, multiple shortest routes are computed from which one or more routes are arbitrarily selected. The wt-ECMP scheme differs from ECMP only in the selection mechanism. Here, all computed shortest routes are assigned a "weight" to ensure that the selection is not arbitrary.

To circumvent the above hurdles and take full advantage of network connectivity, we propose a routing strategy that ensures load balancing, i.e., a strategy that distributes transmission operations as evenly as possible across links. In addition, this approach ensures that no link becomes the network's single potential point of failure.

Two heuristics, referred to as *Load-Balanced, Dynamic, and Replication-aware Routing algorithm* (LB-DRR) and *Congestion Recovering, Dynamic and Replication-aware Routing* (CR-DRR), are proposed in this framework with the following objectives:

- ▷ LB-DRR aims to find a feasible route for each flow so that traffic on each link is minimized. This heuristic also ensures that replicated flows are transmitted on routes that are as disjoint as possible.
- ▷ CR-DRR aims to compute alternative routes for each flow in a runtime congestion situation.

The rest of this chapter is organized as follows. Section 6.1 introduces the model of computation and introduces the notations adopted in this chapter. Our proposed heuristics (LB-DRR and CR-DRR) are explained in more detail in Section 6.2. Finally, Section 6.3 reports on the experiments performed and discussions about them.

6.1 Model of Computation

In this section, we define the network topology and the flow specification that we assume in this chapter. We also introduce the notations and parameters that are necessary for a good understanding of our proposed heuristics.

We note that the assumed model differs from the model presented in Section 4.1 in that the connections here are not full-duplex and each flow may have replicas that are also transmitted over the network. This is because in this chapter we address the routing problem in a more general TSN context and are not limited to preemptive TSN networks only.

▷ **Network topology specification.** We model the network as an undirected graph $\mathcal{G} \stackrel{\text{def}}{=} (\mathcal{V}, \mathcal{E})$, where the set $\mathcal{V} = N \cup \text{SW}_i$ of vertices in $\mathcal{G}$ consists of a finite set N of endpoints and SW_i of switches (see Figure 6.1 for an example). The vertices are connected by a set E of half-duplex links or edges. This means that each edge $e \in E$ is defined by a pair $(v_1, v_2) \in \mathcal{V} \times \mathcal{V}$ of two connected nodes, that allow data to be transmitted in only one direction at a time.

▷ **Flow specification.** We consider a set consisting of n *sporadic* time-sensitive flows $\mathcal{F} \stackrel{\text{def}}{=} \{f_1, f_2, \ldots, f_n\}$. Each flow $f_i \stackrel{\text{def}}{=} (src_i, dst_i, rep_i, C_i, T_i, D_i) \in \mathcal{F}$ is characterized by a

6-tuple, where: (1) src_i is the source node of the flow; (2) dst_i is the destination node of the flow; (3) rep_i is the replication level of the flow (i.e., the number of replicas of f_i that may be transmitted from src_i to dst_i); (4) C_i is the size of the flow; (5) T_i is the minimum inter-arrival time of the flow (also known as the period); and finally, (6) D_i is the deadline of the flow, i.e., the latest time at which at least one copy (original or replicas) of f_i must reach dst_i. We define the set of replicas of f_i as $rep_{f_i} \stackrel{\text{def}}{=} \{f_{i,1}, f_{i,2}, \ldots, f_{i,rep_i}\}$ and assume that each flow and all its replicas are transmitted *simultaneously* over the network.

6.2 Proposed Solution

For the routing problem, we assume that all edges are homogeneous (i.e., they all have the same properties and are interchangeable). Before explaining our proposed routing strategy in detail, we should first define some terms so that the reader can better understand our approach.

Definition 3 (Route)**.** *A* route r_i *of flow* f_i *is defined as an ordered list of edges that can be traversed by* f_i *from its source to its destination. Specifically:*

$$r_i \stackrel{\text{def}}{=} \left\langle (src_i, v_{i,1}), (v_{i,1}, v_{i,2}), \ldots, (v_{i,p}, dst_i) \right\rangle$$

Definition 4 (Valid route)**.** *A* valid route *for* f_i *is defined as any route* r_i *that meets its timing requirement* D_i*.*

Definition 5 (Length of a route)**.** *The length of a route* r_i *denoted by* $len(r_i)$ *is defined as the number of edges along the route.*

Definition 6 (load of an edge)**.** *For every edge* $e = (v_1, v_2) \in \mathcal{V} \times \mathcal{V}$*, we define the* load of e*, denoted by* $load(e)$*, the sum of the sizes of all flows traversing* e*. Formally, the load of edge* e *is defined by Equation 6.1.*

$$load(e) \stackrel{\text{def}}{=} \sum_{f_i \text{ traversing } e} C_i \tag{6.1}$$

Definition 7 (MaxLoad of a route)**.** *The* MaxLoad *of a route* r_i*, denoted by* $Maxload(r_i)$*, is defined as the* maximum *load of all edges in* r_i*. Formally, the MaxLoad of route* r_i *is defined by Equation 6.2.*

$$Maxload(r_i) \stackrel{\text{def}}{=} \max_{e \in r_i} \{load(e)\} \tag{6.2}$$

At this stage, we have all the tools we need to describe our proposed routing solution. The basic idea is as follows. Unlike traditional routing schemes (e.g., SPA, ECMP and wt-ECMP), where the underlying strategy is to find the shortest route for each flow, here we examine all valid routes. If we denote by R_i the set of all valid routes for flow f_i, then our routing strategy is to select the route that leads to the best load distribution in R_i, i.e., the route that minimizes the cost function defined in Equation 6.3.

$$Cost(r_i, K) \stackrel{\text{def}}{=} Maxload(r_i) + K \cdot len(r_i) \tag{6.3}$$

In this Equation 6.3, the parameter $K > 0$ is a user-defined penalty constant. This parameter is meant to penalize the routes with longer length.

> To make it short, parameter K must be considered as a trade-off. It must be set so that the weight of $K \cdot len(r_i)$ in the cost function is significant and $Maxload(r_i)$ does not dominate it, and vice versa.

In the latter case, if $K \cdot len(r_i)$ dominates $Maxload(r_i)$, then the cost function would behave like wt-ECMP. On the other hand, $Maxload(r_i)$ is computed to penalize solutions where some edges in the route transmit a high number of flows, making these edges potential bottlenecks. Last but not least, if multiple routes provide the same lowest cost value, we select one of these routes in an arbitrary manner. Formally, for each flow f_i, its best route $Best(f_i)$ is defined by Equation 6.4.

$$Best(f_i) \stackrel{\text{def}}{=} \min_{r_i \in \text{valid\_routes}} \{Cost(r_i, K)\} \tag{6.4}$$

From this equation, it follows that wt-ECMP is a special case of the proposed approach where the parameter K is sufficiently large and $K \cdot len(r_i)$ dominates over $Maxload(r_i)$. Now we can proceed with the details of our proposed routing schemes.

⋈ **On load-balancing (Algorithm 3).** The load balancing routing scheme (LB-DRR) takes three components as inputs, namely: (1) the network topology $\mathcal{G}$; (2) the set $\mathcal{F}$ of flows to be routed; and finally (3) the user-defined penalty variable K. In the description of the algorithm, the notation $|A|$ refers to the cardinality of the set A.

For each flow f_i, LB-DRR computes the best route after the initialization phase (lines 1 to 3) using the Equation 6.4 (line 6). Then, the load of all edges on this route is updated (line 8) and the selected route is appended to the list of best routes R_i of flow f_i. If the number of replicas of f_i is strictly greater than zero, then all edges already traversed by the original flow f_i are recorded in the variable *used_edge* (line 12). Next, all valid routes

Algorithm 3: LB-DRR.

Data: Network topology $\mathcal{G}$; Set of flows $\mathcal{F}$; Constant K
Result: List of best routes for each flow in F

```
1  R ← empty list[];
2  edges ← Set of all edges in G;
3  load ← zeros[|edges|];
4  foreach f_i ∈ F do
5      R_i ← [];
6      Compute r_i = Best(f_i) (see Equation 6.4);
7      foreach e ∈ r_i do
8          load[e] = load[e] + C_i;
9      end
10     R_i.append(r_i);
11     if rep_i > 0 then
12         used_edges ← {e ∈ r_i};
13         routes = valid_routes(G, src_i, dst_i);
14         for j = 1 to rep_i do
15             r_{i,j} = arg min_{r∈routes} (|used_edges ∩ {e ∈ r}|);
16             foreach egde ∈ r_{i,j} do
17                 load[e] = load[e] + C_i;
18             end
19             used_edges = used_edges ∪ {e ∈ r_{i,j}};
20             R_i.append(r_{i,j});
21         end
22     end
23     R.append(R_i);
24 end
25 return R
```

are computed (line 13) and the route $r_{i,j}$ that has the least overlap with *used_edges* is selected for replica $f_{i,j}$ (with $j \in [1, rep_i]$) (line 15). If multiple routes return the same minimum overlap with *used_edges*, then one of these routes is chosen arbitrarily and the load of all edges is updated to $r_{i,j}$ (line 17). Note that the edges belonging to *used_edges* are also updated to include those traversed by the replica $f_{i,j}$ (line 19). Then, the route $r_{i,j}$ is appended to R_i (line 20) and R_i, the list of selected routes for f_i and its replicas, is appended to the list R of selected routes for all flows (line 23). When this process is complete for all f_i to be transmitted, the algorithm returns the list R (line 25).

▷◁ **On congestion recovery (Algorithm 4).** This algorithm, referred to as CR-DRR, is based on the *Tabu meta-heuristic* (Glover, 1990) and is reactive in that it aims to redirect flows that are in a congestion situation. In short, the main intuition behind any tabu-based

meta-heuristic is to temporarily mark some moves as forbidden in order to force the algorithm to search for alternative solutions that may be better compared to the current solution with respect to a given metric. With this concept in mind, the CR-DRR scheme works as follows. It takes five components as input: (1) the network topology $\mathcal{G}$; (2) the initial routing configuration for all flows R; (3) the congestion threshold $cgst\_threshold$ which defines the upper limit of the allowed load on an edge before it is considered congested; (4) the list of loads on each edge ($load$); and finally (5) the user-defined penalty variable K. All congested edges according to the $cgst\_threshold$ parameter are stored as "tabu-edges" ($csgt\_edges$) and temporarily removed from the network topology (line 2).

For each congestion situation, we initialize the set of congested routes $csgt\_routes$ (i.e., all routes that contain at least one congested edge), the set of flows ($csgt\_flows$) that traverse the congested routes, and the new set of routes R_{new} that contains all routes in R except the congested routes (lines 3 to 5). Then, for each congested flow $f_i \in cgst\_flows$, we search for alternative routes in the new topology (line 7).

From these alternative routes, we choose the best route r_i using Equation 6.3 (line 9). Recall that if multiple routes yield the same minimum cost, we arbitrarily choose one of these routes. We check that the rerouting of the flow f_i does not cause congestion on any edge in r_i (line 10). If this is the case, we leave f_i on its original route old_r_i (line 24). At the end of this process, we update the list $load$ in two phases: (i) on the old route old_r_i: we subtract C_i from all edges (line 12) and (ii) on the new route r_i: we add C_i to all edges (line 15). We update $cgst\_edges$ accordingly (lines 17 to 22). If there is no alternative route for f_i in $New\_Topology$, it stays on its original route old_r_i (line 27). Finally, the computed route is appended to R_{new} (line 29) and when all congested flows have been rerouted, the R_{new} list is returned for all flows (line 31).

Algorithm 4: CR-DRR.

Data: Network topology G; Original routing configuration R; List of loads on each edge ($load$); Congestion threshold $cgst\_threshold$; Constant K

Result: A new routing configuration R_{new}

$cgst\_edges \leftarrow \{e \, in \, \mathcal{G} \,|\, load(e) > cgst\_threshold\}$;
$New\_Topology \leftarrow \mathcal{G} \setminus cgst\_edges$;
$cgst\_routes \leftarrow \{r \in R \,|\, r \cap cgst\_edges \neq \varnothing\}$;
$cgst\_flows \leftarrow \{f_i \, traversing \, a \, route \, in \, cgst\_routes\}$;
$R_{new} \leftarrow R \setminus cgst\_routes$;
foreach $f_i \in cgst\_flows$ **do**
    $routes \leftarrow valid\_routes(New\_Topology, src_i, dst_i)$;
    if $(routes \neq \varnothing)$ **then**
        $r_i = \arg\min_{r \in routes}(Cost(r, K))$;
        if $(Maxload(r_i) \leq cgst\_threshold)$ **then**
            foreach $e \in old\_r_i$ **do**
                $load[e] = load[e] - C_i$;
            end
            foreach $e \in r_i$ **do**
                $load[e] = load[e] + C_i$;
            end
            foreach $e \in cgst\_edges$ **do**
                if $(load(e) \leq cgst\_threshold)$ **then**
                    $New\_Topology = New\_Topology.add(e)$;
                    $cgst\_edges = cgst\_edges \setminus \{e\}$;
                end
            end
        else
            $r_i = old\_r_i$
        end
    else
        $r_i = old\_r_i$
    end
    $R_{new}.add(r_i)$;
end
return R_{new}

6.3 Experimental Evaluation

In this section, we report on the experiments we performed on synthetic workloads to evaluate the performance of the proposed heuristics (LB-DRR and CR-DRR) in terms of the maximum load transmitted on an edge compared to SPA and wt-ECMP. We then evaluate the scalability of the proposed algorithms to demonstrate their applicability.

▷ **Setup**. We consider a TSN network with 50 nodes and a connectivity degree falling in the interval $[0.15, 0.35]$, modeled as an *Erdős-Rényi* (ER) graph (Erdős et al., 2013). Such a graph can be generated by starting with a set of n nodes and independently adding edges between them with probability p. The result is a graph with a random number of edges. The expected number of edges in an ER graph is equal to $n(n-1)/2 \times p$. Here p is our predefined connectivity degree. We set K = 100 and randomly generate up to 1000 real-time flows in window $[25, 200]$, where the size of each flow is between 200 and 1000 bytes, the deadlines are between 2 *ms* and 5 *ms*, and the replication level between 0 and 2. In the first set of experiments, we adopted the LB-DRR routing scheme, and in the second set, we adopted the SPA routing scheme. In the latter case, we applied the algorithm CR-DRR to reroute the flows in case of congestion.

▷ **Results and discussion.** In the first set of experiments, we found that LB-DRR reduces the maximum load transmitted on an edge (Maxload) by an average of 70.3% and 23.3%, respectively, compared to SPA and wt-ECMP. Figure 6.2 shows the Maxload for each routing scheme when the number of flows varies and LB-DRR clearly dominates both SPA and wt-ECMP.

In Figure 6.3, we observed that LB-DRR performs better by varying the connectivity level of the network and its Maxload decreases significantly.

Note that higher connectivity leads to longer runtime overhead due to the increasing number of routes to consider. Figure 6.4 illustrates the scalability of LB-DRR as the number of flows increase.

As for increasing the number of flows, we found that LB-DRR scales linearly, but very slowly (it took only 26 seconds to compute routes for 1000 flows). Now, if we increase the number of nodes, we set the connectivity level of the network to 0.2 and consider 100 real-time flows. Figure 6.5 shows that the execution time of LB-DRR increases exponentially when the number of nodes exceeds 75. However, it could still compute routes for 125 nodes in 11 minutes.

In the second set of experiments, we routed 750 real-time flows with *SPA*, and observed tremendous congestion on the selected routes. Figures 6.6 and 6.7 show the results of congestion recovery and load redistribution.

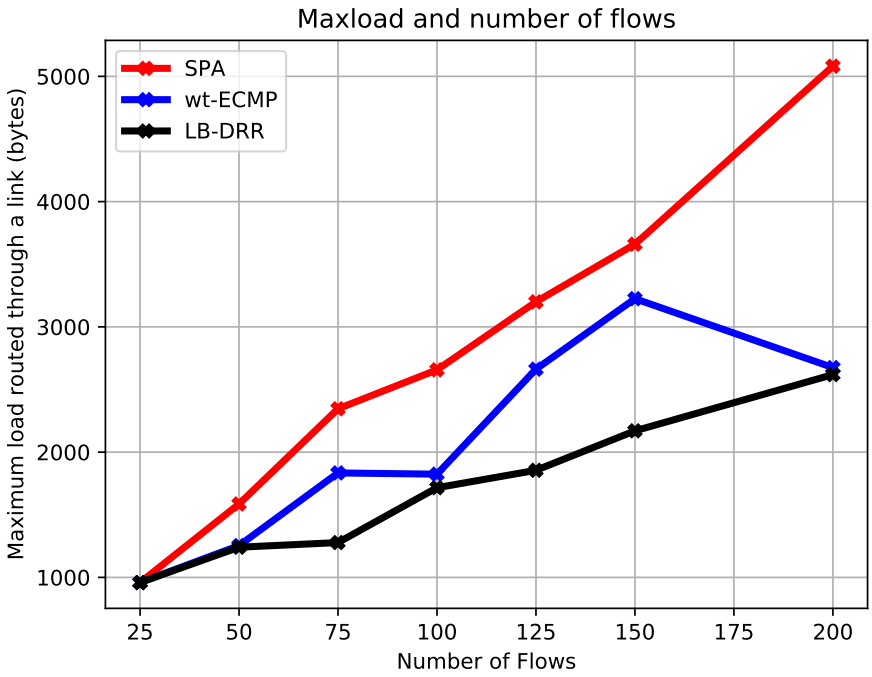


Figure 6.2: Load balancing: LB-DRR vs. SPA and wt-ECMP.

In Figure 6.6, the network load was initially unbalanced (see the red curve), with multiple flows routed only a limited number of edges (see the leftmost peak), while several edges remained unused (see the rightmost long end). By applying the CR-DRR scheme, a significant improvement was observed (see the black curve). Figure 6.7 shows the load distribution of the congested network before and after applying CR-DRR.

From Figure 6.7, it can be seen that the load distribution curve of CR-DRR is close to the normal distribution. Finally, CR-DRR shows the same behavior as LB-DRR in terms of scalability.

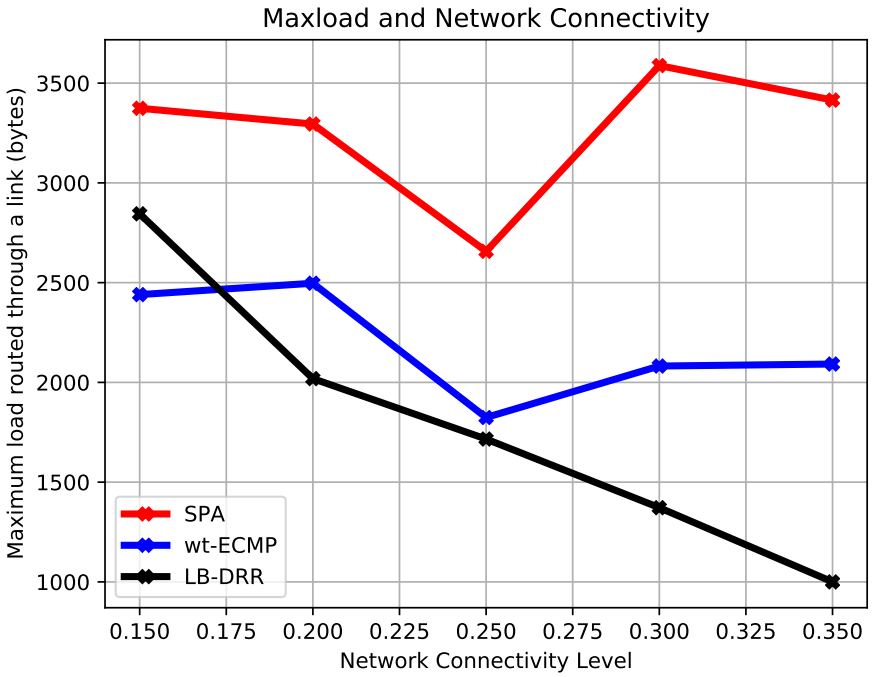


Figure 6.3: Performance improvement w.r.t. network connectivity.

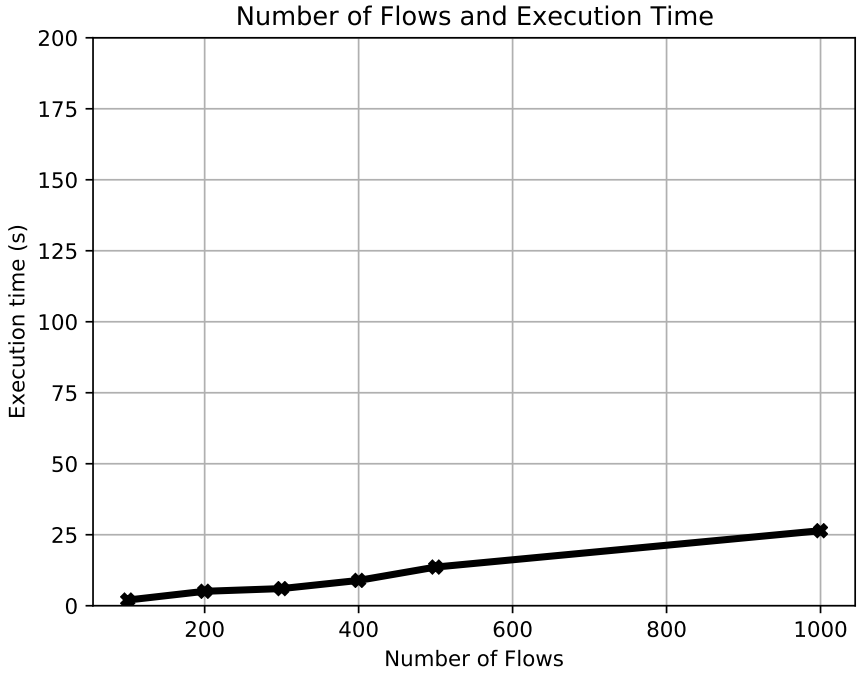


Figure 6.4: Scalability w.r.t. number of flows.

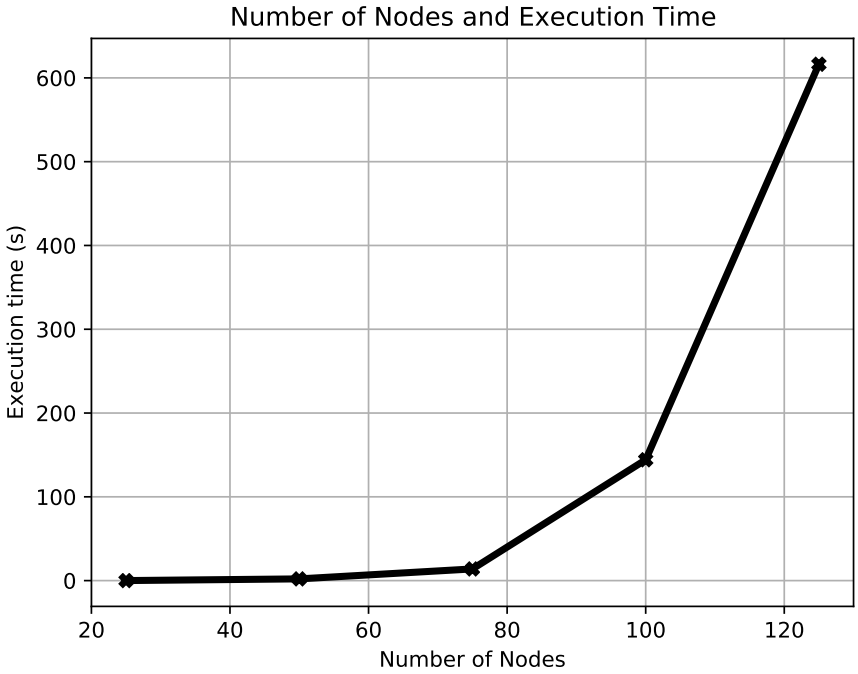


Figure 6.5: Scalability w.r.t. number of nodes.

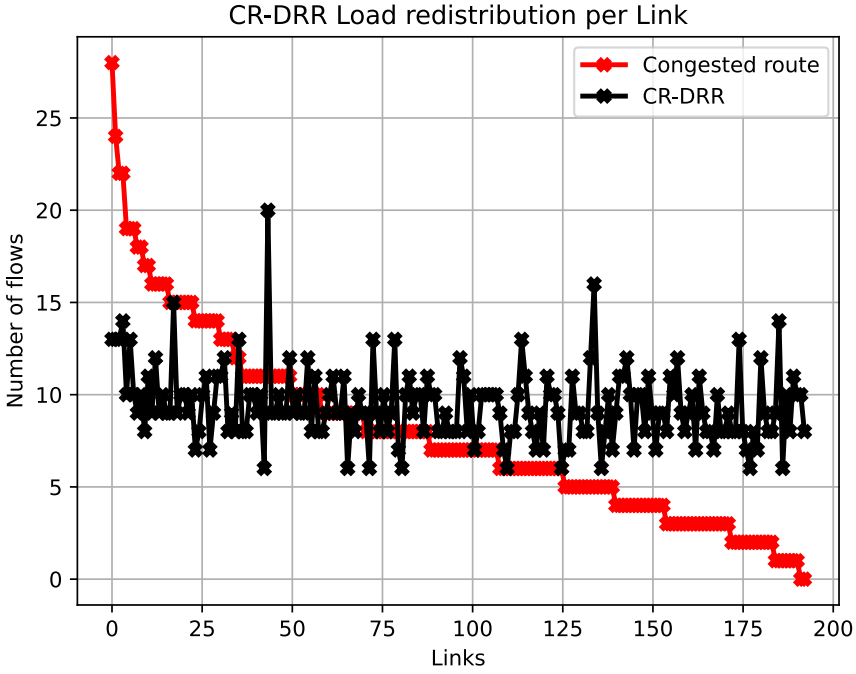


Figure 6.6: CR-DRR adopted to recover from congestion

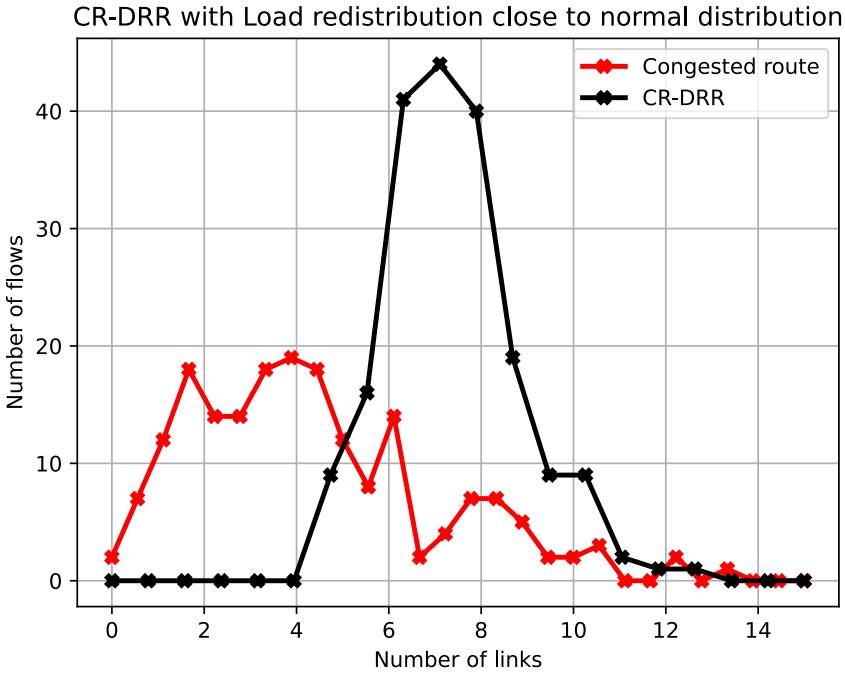


Figure 6.7: Load distribution under CR-DRR.

Chapter Summary

In this chapter, we proposed two routing heuristics, referred to as LB-DRR and CR-DRR, to address the problems of load-balancing and congestion in TSN networks. We evaluated the performance of the proposed schemes against the popular SPA and wt-ECMP routing algorithms and showed an improvement of more than 70% and 20%, respectively. This improvement has been observed w.r.t. the maximum load transmitted on an edge. On another front, the proposed heuristics exhibit high scalability w.r.t. an increase in the number of flows.

Chapter 7

Conclusions

In this chapter, we summarize the findings and results of this thesis and outline possible future research directions. In particular, we summarize our contributions and draw some conclusions in Section 7.1. In Section 7.2 we examine the validity of the thesis statement, and finally, we outline the limitations of the work and possible areas for improvement in Section 7.3.

7.1 Summary

To achieve low latency in the transmission of time-sensitive flows in Ethernet networks, the IEEE has introduced the IEEE 802.1Qbu standard, which specifies a 1-level preemption scheme. Here, frames are divided into two different preemption classes — namely, *express frames* and *preemptable frames* — and a so-called preemptable frame can be interrupted before it is completed to allow fast transmission of *express frames*; but any other preemptable frame cannot be transmitted until the already preempted frame is completed. While this approach improves the responsiveness of express frames, their performance is negatively affected as the number increases. In addition, there are a number of preemptable frames that cannot be promoted as express frames, but for which firm timing constraints apply. These frames may experience long blocking times because two frames belonging to the same preemption class cannot preempt each other. To overcome these limitations, we have proposed a multi-level preemption scheme in this work.

We first provided a detailed background and literature review on real-time communication in the context of Distributed Real-time Embedded Systems (DRES) in Chapter 2. Then, we examined the preemption mechanism in detail to determine the feasibility of a multi-level preemption scheme, and our analysis confirms its feasibility. In Chapter 3, we detailed the changes required to achieve this goal and provided implementation

recommendations to ensure frame integrity and interoperability. We then compared the hardware implementation costs of the multi-level preemption scheme with TAS and CBS. Then, a formal timing guarantee for each flow under such a scheme was given using a so-called *Compositional Performance Analysis* in Chapter 4. In addition, we presented a configuration framework for determining the appropriate preemption level to enable, as well as a methodology for assigning flows to preemption classes for the preemption scheme. We extensively evaluated the performance of the proposed muti-level preemption scheme as well as the configuration framework in Chapter 5. Finally, we studied traffic routing in the general context of TSN networks in Chapter 6. Here, we propose two routing heuristics, called LB-DRR and CR-DRR, to solve the problem of congestion avoidance and congestion recovery, respectively.

7.2 Thesis validation

In Chapter 1, we stated the thesis supported by this dissertation as follows:

> *We postulate that multi-level preemption when added to TSN can mitigate several shortcomings of this standard and unlock benefits in timeliness and resource utilization of DRES.*

In this dissertation, we have shown that multi-level preemption leads to substantial improvements in performance. In particular, we have shown in Chapter 5 that multi-level preemption leads to an improvement of up to 53.07% in WCTT guarantees for preemptable time-sensitive frames with firm timing constraints, and that the average number of schedulable flowsets increases with increasing preemption levels. In an extensive space exploration experiment, the schedulability ratio jumps from 45.5% under the non-preemptive scheme to 85.8% under the 1-level preemption scheme, and the trend continues, albeit non-linearly, with each additional preemption level to peak at 98.1% under a fully preemptive scheme.

From these observations, it follows that the constraints of the multi-level preemption mitigate the shortcomings of 1-level preemption as defined in the standards and unlock benefits in terms of timeliness and resource utilization. Therefore, the thesis is validated.

It is important to note that each preemption level introduces additional hardware implementation overheads and that the framework presented in chapter 4 is important to

ensure that only the required preemption levels are enabled and hardware resources are conserved.

> We also note that the improvement from additional preemptions is non-linear and drops off sharply after 4 preemption levels. Interestingly, this (4) is also the point after which the implementation overheads of the multi-level preemption scheme overtake that of TAS, as shown by the results in Chapter 3.

Recent studies have shown that the performance of standard Ethernet with Frame Preemption is comparable to that of widely studied time-aware shapers (TAS), which are more complex and expensive to implement. By addressing the key limitations of the 1-level preemption scheme, this work provides a simpler and less expensive alternative communication solution.

7.3 Limitations and future directions

In this section, we acknowledge the limitations of our work and suggest ways in which it could be improved and extended. We believe that this topic can be further explored and developed, and we encourage future research in this direction to accelerate the adoption of multi-level preemption systems in DRES. Below, we highlight the limitations and potential areas for improvement.

7.3.1 Hardware implementation

In this work, we have provided a detailed description of the required changes to the switch transmission and reception mechanisms to support multi-level preemption and the implication in terms of hardware overhead. However, due to resource constraints, we did not implement actual proof-of-concept switch hardware. Implementation costs were estimated using the resource utilization profile of the 1-level preemption scheme, assuming that this utilization increases linearly with each additional level of preemption. This assumption is based on our proposed implementation approach, which requires an additional MAC merge sublayer for each additional level of preemption. We chose this approach because it ensures interoperability with other preemption schemes (non-preemptive and 1- level) and does not require any change to the Ethernet frame format. We note that another approach has been proposed in the literature by Knezic et al. (2020). This approach promises an implementation with lower hardware overhead but requires modification of the standard Ethernet frame format, which may affect interoperability.

The development of an actual proof-of-concept switch that supports multi-level preemption is an interesting direction for the future. In addition, studies on more efficient implementation approaches would greatly benefit multi-level preemption scheme.

7.3.2 Time-triggered and reservation-based models

To analyze the timing properties of the multi-level preemption scheme, this work assumes only Ethernet with Strict Priority and Frame Preemption. We choose this model so that we can focus solely on evaluating multi-level preemption without the added complexity of other protocol mechanisms.

Further studies on the impact of multi-level preemption on time-triggered and reservation-based transmission schemes could be conducted to evaluate how these mechanisms can benefit from multi-level preemption. Another interesting research direction would be to compare the performance of strict priority and multi-level preemption with the time-triggered and reservation-based TSN traffic control mechanisms from WCTT and QoS perspectives.

7.3.3 Starvation

In strict priority preemptive networks, lower priority flows are always at risk of starvation. We have not addressed this challenge in this dissertation. We believe that CBS is a promising candidate to protect frames of lower priority classes from starvation. Another promising concept that can be helpful in this regard is the so-called "aging" where the priority of a flow (or task) increases with respect to the amount of time that it has spent in the network (or system) (Buttazzo, 2011).

We strongly believe that investigating novel methods to mitigate starvation in multi-level preemption schemes is a promising direction.

Bibliography

T. Ae and R. Aibara, "Programmable real-time scheduler using a neurocomputer," *Real-Time Systems*, vol. 1, no. 4, pp. 351–363, 1990. [Online]. Available: https://doi.org/10.1007/BF00366575 Cited on page 50.

Airbus, "Aeronautical radio inc.,," *ARINC specification 664P7: aircraft data network, part 7*, 2006. [Online]. Available: https://aviation-ia.sae-itc.com/standards/arinc664p7-1-664p7-1-aircraft-data-network-part-7-avionics-full-duplex-switched-ethernet-network

B. Akesson, M. Nasri, G. Nelissen, S. Altmeyer, and R. I. Davis, "An empirical survey-based study into industry practice in real-time systems," in *IEEE Real-Time Systems Symposium (RTSS)*, 2020, pp. 3–11. [Online]. Available: https://doi.org/10.1109/RTSS49844.2020.00012 Cited on page 14.

G. Alderisi, G. Iannizzotto, and L. L. Bello, "Towards IEEE 802.1 ethernet avb for advanced driver assistance systems: A preliminary assessment," in *17th International Conference on Emerging Technologies & Factory Automation (ETFA)*, 2012, pp. 1–4. [Online]. Available: https://doi.org/10.1109/ETFA.2012.6489775 Cited on page 100.

G. Alderisi, G. Patti, and L. L. Bello, "Introducing support for scheduled traffic over IEEE audio video bridging networks," in *IEEE 18th Conference on Emerging Technologies & Factory Automation (ETFA)*, 2013, pp. 1–9. [Online]. Available: https://doi.org/10.1109/ETFA.2013.6647943 Cited on page 46.

R. Alvizu, G. Maier, N. Kukreja, A. Pattavina, R. Morro, A. Capello, and C. Cavazzoni, "Comprehensive survey on t-sdn: Software-defined networking for transport networks," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2232–2283, 2017. [Online]. Available: https://doi.org/10.1109/COMST.2017.2715220

C. Aras, J. Kurose, D. Reeves, and H. Schulzrinne, "Real-time communication in packet-switched networks," *Proceedings of the IEEE*, vol. 82, no. 1, pp. 122–139, 1994. [Online]. Available: https://doi.org/10.1109/5.259431 Cited on pages 17 and 18.

F. A. R. Arif and T. S. Atia, "Load balancing routing in time-sensitive networks," in *Third International Scientific-Practical Conference Problems of Infocommunications Science and Technology (PIC S&T)*, 2016, pp. 207–208. [Online]. Available: https://doi.org/10.1109/INFOCOMMST.2016.7905383 Cited on page 50.

M. Ashjaei, G. Patti, M. Behnam, T. Nolte, G. Alderisi, and L. Lo, "Schedulability analysis of Ethernet Audio Video Bridging networks with scheduled traffic support," *Real-Time Systems*, vol. 53, no. 4, pp. 526–577, 2017. [Online]. Available: https://doi.org/10.1007/s11241-017-9268-5

M. Ashjaei, M. Sjödin, and S. Mubeen, "A novel frame preemption model in tsn networks," *Journal of Systems Architecture*, vol. 116, p. 102037, 2021. [Online]. Available: https://doi.org/10.1016/j.sysarc.2021.102037 Cited on pages 5 and 14.

N. Audsley, "On priority assignment in fixed priority scheduling," *Information Processing Letters*, vol. 79, no. 1, pp. 39–44, 2001. [Online]. Available: https://doi.org/10.1016/S0020-0190(00)00165-4 Cited on page 49.

P. Axer, D. Thiele, R. Ernst, and J. Diemer, "Exploiting shaper context to improve performance bounds of ethernet avb networks," in *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2014, pp. 1–6. [Online]. Available: https://doi.org/10.1145/2593069.2593136

H. Bauer, J.-L. Scharbarg, and C. Fraboul, "Applying trajectory approach with static priority queuing for improving the use of available afdx resources," *Real-Time Systems*, vol. 48, no. 1, p. 101–133, jan 2012. [Online]. Available: https://doi.org/10.1007/s11241-011-9142-9 Cited on page 47.

S. Beji, S. Hamadou, A. Gherbi, and J. Mullins, "Smt-based cost optimization approach for the integration of avionic functions in ima and ttethernet architectures," in *IEEE/ACM 18th International Symposium on Distributed Simulation and Real Time Applications*, 2014, pp. 165–174. [Online]. Available: https://doi.org/10.1109/DS-RT.2014.28 Cited on page 49.

L. L. Bello, "Novel trends in automotive networks: A perspective on ethernet and the IEEE audio video bridging," in *IEEE Emerging Technology and Factory Automation (ETFA)*, 2014, pp. 1–8. [Online]. Available: https://doi.org/10.1109/ETFA.2014.7005251 Cited on page 46.

G. Berardinelli and H. Viswanathan, "Overlay transmission of sporadic random access and broadband traffic for 5g networks," in *International Symposium on Wireless Communication Systems (ISWCS)*, 2017, pp. 19–24. [Online]. Available: https://doi.org/10.1109/ISWCS.2017.8108108

W. D. Blizard *et al.*, "Multiset theory," *Notre Dame Journal of formal logic*, vol. 30, no. 1, pp. 36–66, 1989. Cited on page 91.

D. R. Boggs, J. C. Mogul, and C. A. Kent, "Measured capacity of an ethernet: Myths and reality," in *Symposium Proceedings on Communications Architectures and Protocols*, ser. SIGCOMM '88. New York, NY, USA: Association for Computing Machinery, 1988, p. 222–234. [Online]. Available: https://doi.org/10.1145/52324.52347

D. R. Boggs, J. C. Mogul, and C. A. Kent, *Measured Capacity of an Ethernet: Myths and Reality*. New York, NY, USA: Association for Computing Machinery, aug 1988, vol. 18, no. 4. [Online]. Available: https://doi.org/10.1145/52325.52347

F. Brajou and P. Ricco, "The airbus a380 - an afdx-based flight test computer concept," in *Proceedings AUTOTESTCON 2004.*, 2004, pp. 460–463. [Online]. Available: https://doi.org/10.1109/AUTEST.2004.1436931 Cited on page 2.

S. Brooks and E. Uludag, "Time-sensitive networking: From theory to implementation in industrial automation," *White Paper, Time Sensitive Networking, Intel*, 2019. [Online]. Available: https://www.intel.com/content/dam/support/us/en/programmable/support-resources/bulk-container/pdfs/literature/wp/wp-01279-time-sensitive-networking-from-theory-to-implementation-in-industrial-automation.pdf

S. D. Brown and Z. G. Vranesic, *Fundamentals of Digital Logic with Verilog Design*, 3rd ed. New York: McGraw-Hill Higher Education, 2014. Cited on pages 60 and 61.

A. Burns, K. Tindell, and A. Wellings, "Effective analysis for engineering real-time fixed priority schedulers," *IEEE Transaction on Software Engineering*, vol. 21, no. 5, p. 475–480, may 1995. [Online]. Available: https://doi.org/10.1109/32.387477

G. C. Buttazzo, *Hard Real-Time Computing Systems - Predictable Scheduling Algorithms and Applications*, 3rd ed. Springer, 2011. [Online]. Available: https://doi.org/10.1007/978-1-4614-0676-1 Cited on page 136.

D. Buttle, "Real-time in the prime-time, etas gmbh, germany," in *Keynote talk at 24th Euromicro Conference on Real-Time Systems, Pisa, Italy*, 2012. Cited on page 49.

J. Cao, P. J. L. Cuijpers, R. J. Bril, and J. J. Lukkien, "Tight worst-case response-time analysis for ethernet avb using eligible intervals," in *IEEE World Conference on Factory Communication Systems (WFCS)*, 2016, pp. 1–8. [Online]. Available: https://doi.org/10.1109/WFCS.2016.7496507 Cited on page 47.

J. Cao, P. J. Cuijpers, R. J. Bril, and J. J. Lukkien, "Independent yet tight wcrt analysis for individual priority classes in ethernet avb," in *Proceedings of the 24th International Conference on Real-Time Networks and Systems*, ser. RTNS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 55–64. [Online]. Available: https://doi.org/10.1145/2997465.2997493 Cited on page 48.

C. Cardeira and Z. Mammeri, "Neural networks for multiprocessor real-time scheduling," in *6th Euromicro Workshop on Real-Time Systems*, 1994, pp. 59–64. [Online]. Available: https://doi.org/10.1109/EMWRTS.1994.336864 Cited on page 50.

C. Cardeira and Z. Mammeri, "Preemptive and non-preemptive real-time scheduling based on neural networks," in *13th IFAC Workshop on Distributed Computer Control Systems 1995(DCCS '95)*, 1995, vol. 28, no. 22, pp. 67–72. [Online]. Available: https://doi.org/10.1016/S1474-6670(17)46670-5 Cited on page 50.

H. Charara, J.-L. Scharbarg, J. Ermont, and C. Fraboul, "Methods for bounding end-to-end delays on an afdx network," in *18th Euromicro Conference on Real-Time Systems (ECRTS'06)*, 2006, pp. 10 pp.–202. [Online]. Available: https://doi.org/10.1109/ECRTS.2006.15

F. Cottet, J. Delacroix, C. Kaiser, and Z. Mammeri, *Scheduling in real-time systems*. Computing Reviews, 2004, vol. 45, no. 12. Cited on pages 18 and 20.

S. S. Craciunas, R. S. Oliver, M. Chmelík, and W. Steiner, "Scheduling real-time communication in ieee 802.1 qbv time sensitive networks," in *24th International Conference on Real-Time Networks and Systems (RTNS)*, 2016, pp. 183–192. [Online]. Available: https://doi.org/10.1145/2997465.2997470

S. S. Craciunas, R. S. Oliver, and W. Steiner, "Formal scheduling constraints for time-sensitive networks," *arXiv preprint arXiv:1712.02246*, 2017. [Online]. Available: https://doi.org/10.48550/arXiv.1712.02246

P. Danielis, J. Skodzik, V. Altmann, E. B. Schweissguth, F. Golatowski, D. Timmermann, and J. Schacht, "Survey on real-time communication via ethernet in industrial automation environments," in *IEEE Emerging Technology and Factory Automation (ETFA)*, 2014, pp. 1–8. [Online]. Available: https://doi.org/10.1109/ETFA.2014.7005074

R. I. Davis and A. Burns, "Robust priority assignment for messages on controller area network (CAN)," *Real-Time Systems*, vol. 41, no. 2, pp. 152–180, 2009. [Online]. Available: https://doi.org/10.1007/s11241-008-9065-2 Cited on page 49.

R. I. Davis, A. Burns, R. J. Bril, and J. J. Lukkien, "Controller area network (can) schedulability analysis: Refuted, revisited and revised," *Real-Time Systems*, vol. 35, no. 3, pp. 239–272, 2007. [Online]. Available: https://doi.org/10.1007/s11241-007-9012-7 Cited on pages 19 and 88.

R. I. Davis, L. Cucu-Grosjean, M. Bertogna, and A. Burns, "A review of priority assignment in real-time systems," *J. Syst. Archit.*, vol. 65, no. C, p. 64–82, 2016. [Online]. Available: 10.1016/j.sysarc.2016.04.002 Cited on pages 48, 49, 88, and 110.

J. Diemer, P. Axer, and R. Ernst, "Compositional performance analysis in python with pycpa," in *3rd International Workshop on Analysis Tools and Methodologies for Embedded Real-TIme Systems (WATERS)*, 2012, pp. 27–32.

J. Diemer, D. Thiele, and R. Ernst, "Formal worst-case timing analysis of ethernet topologies with strict-priority and avb switching," in *7th IEEE International Symposium on Industrial Embedded Systems (SIES'12)*, 2012, pp. 1–10. [Online]. Available: https://doi.org/10.1109/SIES.2012.6356564 Cited on pages 48 and 76.

F. Dürr and N. G. Nayak, "No-wait packet scheduling for IEEE time-sensitive networks (tsn)," in *24th International Conference on Real-Time Networks and Systems*, ser. RTNS '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 203–212. [Online]. Available: https://doi.org/10.1145/2997465.2997494

L. Erdős, A. Knowles, H.-T. Yau, J. Yin *et al.*, "Spectral statistics of erdős–rényi graphs i: local semicircle law," *The Annals of Probability*, vol. 41, no. 3B, pp. 2279–2375, 2013. [Online]. Available: https://doi.org/10.48550/arXiv.1103.1919 Cited on page 127.

J. Falk, D. Hellmanns, B. Carabelli, N. Nayak, F. Dürr, S. Kehrer, and K. Rothermel, "Nesting: Simulating IEEE time-sensitive networking (tsn) in omnet++," in *2019 International Conference on Networked Systems (NetSys)*, 2019, pp. 1–8. [Online]. Available: https://doi.org/10.1109/NetSys.2019.8854500 Cited on pages 12 and 100.

J. Farkas, N. Finn, and P. Thaler, "Networking (TSN) Before We Start," pp. 1–55, 2017.

M. H. Farzaneh and A. Knoll, "Time-sensitive networking (tsn): An experimental setup," in *IEEE Vehicular Networking Conference (VNC)*, 2017, pp. 23–26. [Online]. Available: https://doi.org/10.1109/VNC.2017.8275648

M. H. Farzaneh, S. Shafaei, and A. Knoll, "Formally verifiable modeling of in-vehicle time-sensitive networks (tsn) based on logic programming," in *IEEE Vehicular Networking Conference (VNC)*, 2016, pp. 1–4. [Online]. Available: https://doi.org/10.1109/VNC.2016.7835941

J. Feld, "Profinet - scalable factory communication for all applications," in *IEEE International Workshop on Factory Communication Systems*, 2004, pp. 33–38. [Online]. Available: https://doi.org/10.1109/WFCS.2004.1377673 Cited on pages 2 and 27.

N. Finn, "Time-sensitive and Deterministic Networking Whitepaper," pp. 1–24, 2017. [Online]. Available: https://mentor.ieee.org/802.24/dcn/17/24-17-0020-00-sgtg-contribution-time-sensitive-and-deterministic-networking-whitepaper.pdf Cited on page 77.

N. Finn, "IEEE draft standard for local and metropolitan area networks —link-local registration protocol," *IEEE P802.1CS/D2.2*, 2019. [Online]. Available: https://1.ieee802.org/tsn/802-1cs/ Cited on page 31.

F. Frances, C. Fraboul, and J. Grieu, "Using network calculus to optimize the AFDX network," in *3rd European Congress on Embedded Real Time Software (ERTS)*, 2006. [Online]. Available: https://hal.archives-ouvertes.fr/hal-02270458

V. Gavriluţ and P. Pop, "Traffic-type assignment for tsn-based mixed-criticality cyber-physical systems," *ACM Transaction on Cyber-Physical Systems*, vol. 4, no. 2, jan 2020. [Online]. Available: https://doi.org/10.1145/3371708 Cited on pages xiv, 49, 114, and 115.

V. Gavrilut, B. Zarrin, P. Pop, and S. Samii, "Fault-tolerant topology and routing synthesis for IEEE time-sensitive networking," in *25th International Conference on Real-Time Networks and Systems*, ser. RTNS '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 267–276. [Online]. Available: https://doi.org/10.1145/3139258.3139284 Cited on pages 13, 51, and 119.

V. Gavriluţ and P. Pop, "Scheduling in time sensitive networks (tsn) for mixed-criticality industrial applications," in *14th IEEE International Workshop on Factory Communication Systems (WFCS)*, 2018, pp. 1–4. [Online]. Available: https://doi.org/10.1109/WFCS.2018.8402374

F. Glover, "Tabu search: A tutorial," *Interfaces*, vol. 20, no. 4, pp. 74–94, 1990. [Online]. Available: https://doi.org/10.1287/inte.20.4.74 Cited on page 124.

A. Gogolev and P. Bauer, "A simpler tsn? traffic scheduling vs. preemption." in *25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, 2020, pp. 183–189. [Online]. Available: https://doi.org/10.1109/ETFA46521.2020.9211987 Cited on pages 5, 6, 14, 44, 46, and 59.

A. Gogolev and R. Braun, "End system tsn enablement using opc ua," in *International Conference on Emerging Technologies and Factory Automation*, 2021, pp. 01–07. [Online]. Available: https://doi.org/10.1109/ETFA45728.2021.9613458

M. D. Grammatikakis, D. Hsu, M. Kraetzl, and J. F. Sibeyn, "Packet routing in fixed-connection networks: A survey," *Journal of Parallel and Distributed Computing*, vol. 54, no. 2, pp. 77–132, 1998. [Online]. Available: https://doi.org/10.1006/jpdc.1998.1483 Cited on page 50.

A. Grigorjew, F. Metzger, T. Hoßfeld, J. Specht, F.-J. Götz, F. Chen, and J. Schmitt, "Asynchronous traffic shaping with jitter control," Universität Würzburg, Tech. Rep., 2020. [Online]. Available: https://doi.org/https://doi.org/10.25972/OPUS-20582

E. Grossman, "Deterministic networking use cases," *RFC 8578*, no. 8578, May 2019. [Online]. Available: https://doi.org/10.17487/RFC8578

D. B. Gustavson, "Computer buses a tutorial," *IEEE Micro*, vol. 4, no. 4, pp. 7–22, 1984. [Online]. Available: https://doi.org/10.1109/MM.1984.291216 Cited on page 19.

M. Gutiérrez, W. Steiner, R. Dobrin, and S. Punnekkat, "A configuration agent based on the time-triggered paradigm for real-time networks," in *IEEE World Conference on Factory Communication Systems (WFCS)*, 2015, pp. 1–4. [Online]. Available: https://doi.org/10.1109/WFCS.2015.7160584

M. Gutiérrez, W. Steiner, R. Dobrin, and S. Punnekkat, "Learning the parameters of periodic traffic based on network measurements," in *IEEE International Workshop on Measurements & Networking (M&N)*, 2015, pp. 1–6. [Online]. Available: https://doi.org/10.1109/IWMN.2015.7322981

M. Gutiérrez, A. Ademaj, W. Steiner, R. Dobrin, and S. Punnekkat, "Self-configuration of IEEE 802.1 tsn networks," in *22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2017, pp. 1–8. [Online]. Available: https://doi.org/10.1109/ETFA.2017.8247597

E. Haleplidis, K. Pentikousis, S. Denazis, J. H. Salim, D. Meyer, and O. Koufopavlou, "Software-Defined Networking (SDN): Layers and Architecture Terminology," RFC 7426, Jan. 2015. [Online]. Available: 10.17487/RFC7426

F. Hanssen and P. G. Jansen, *Real-time communication protocols: an overview*. Citeseer, 2003. [Online]. Available: https://ris.utwente.nl/ws/portalfiles/portal/5151857/hanssen.pdf

T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning*. New York, USA: Springer, 2009. [Online]. Available: https://doi.org/10.1007/978-0-387-84858-7 Cited on pages 91 and 92.

P. Heise, F. Geyer, and R. Obermaisser, "Tsimnet: An industrial time sensitive networking simulation framework based on omnet++," in *8th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, 2016, pp. 1–5. [Online]. Available: https://doi.org/10.1109/NTMS.2016.7792488

D. Hellmanns, J. Falk, A. Glavackij, R. Hummen, S. Kehrer, and F. Dürr, "On the performance of stream-based, class-based time-aware shaping and frame preemption in tsn," in *IEEE International Conference on Industrial Technology (ICIT)*, 2020, pp. 298–303. [Online]. Available: https://doi.org/10.1109/ICIT45562.2020.9067122 Cited on page 59.

R. Henia, A. Hamann, M. Jersak, R. Racu, K. Richter, and R. Ernst, "System level performance analysis–the symta/s approach," *Computers and Digital Techniques*, vol. 152, no. 2, pp. 148–166, 2005. [Online]. Available: https://doi.org/10.1049/ip-cdt:20045088 Cited on pages 47 and 74.

T. Herpel, B. Kloiber, R. German, and S. Fey, "Routing of safety-relevant messages in automotive ecu networks," in *IEEE 70th Vehicular Technology Conference Fall*, 2009, pp. 1–5. [Online]. Available: https://doi.org/10.1109/VETECF.2009.5378778

R. Hofmann, L. Ahrendts, and R. Ernst, "Cpa: Compositional performance analysis," in *Handbook of Hardware/Software Codesign*, S. Ha and J. Teich, Eds. Dordrecht: Springer Netherlands, 2017, pp. 721–751. [Online]. Available: https://doi.org/10.1007/978-94-017-7267-9_24 Cited on pages 48, 74, and 78.

Y. Hotta, A. Inoue, H. Bessho, C. Mangin, and R. Kawate, "Experimental study of a low-delay ethernet switch for real time networks," 2015. [Online]. Available: https://www.mitsubishielectric-rce.eu/wp-content/uploads/2020/02/Mangin_IEEE2015.pdf Cited on page 45.

M. Huang, E. Miller, P. Sun, and C. Oji, "Ethernet: An engineering paradigm," *Semantic Scholar [online]*, 1998. Cited on page 23.

I. IEC, "61158-1 industrial communication networks: fieldbus specifications," 2007. Cited on page 26.

IEEE, "IEEE standard for local and metropolitan area network–bridges and bridged networks," *IEEE Std 802.1Q-2018 (Revision of IEEE Std 802.1Q-2014)*, pp. 1–1993, 2018. [Online]. Available: https://doi.org/10.1109/IEEESTD.2018.8403927 Cited on pages 4, 63, 77, and 79.

IEEE, "Ieee standard for a precision clock synchronization protocol for networked measurement and control systems," *IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002)*, pp. 1–269, 2008. [Online]. Available: https://doi.org/10.1109/IEEESTD.2008.4579760 Cited on page 30.

IEEE, "IEEE standard for local and metropolitan area networks–audio video bridging (avb) systems," *IEEE Std 802.1BA-2011*, pp. 1–45, 2011. [Online]. Available: https://doi.org/10.1109/IEEESTD.2011.6032690 Cited on pages 28 and 48.

IEEE, "Ieee standard for ethernet amendment 5: Specification and management parameters for interspersing express traffic," *IEEE Std 802.3br-2016 (Amendment to IEEE Std 802.3-2015 as amended by IEEE St802.3bw-2015, IEEE Std 802.3by-2016, IEEE Std 802.3bq-2016, and IEEE Std 802.3bp-2016)*, pp. 1–58, 2016. [Online]. Available: https://doi.org/10.1109/IEEESTD.2016.7900321

IEEE, "Bridges and Bridged Networks — Amendment : YANG Data MOdel," 2017. [Online]. Available: http://www.IEEE802.org/1/files/private/cp-drafts/d2/802-1Qcp-d2-0.pdf

IEEE, "IEEE standard for local and metropolitan area networks–bridges and bridged networks - amendment 34:asynchronous traffic shaping," *IEEE Std 802.1Qcr-2020 (Amendment to IEEE Std 802.1Q-2018 as amended by IEEE Std 802.1Qcp-2018, IEEE Std 802.1Qcc-2018, IEEE Std 802.1Qcy-2019, and IEEE Std 802.1Qcx-2020)*, pp. 1–151, 2020. [Online]. Available: https://doi.org/10.1109/IEEESTD.2020.9253013

IEEE, "Ieee standard for local and metropolitan area networks - timing and synchronization for time-sensitive applications in bridged local area networks," *IEEE Std 802.1AS-2011*, pp. 1–292, 2011. [Online]. Available: https://doi.org/10.1109/IEEESTD.2011.5741898 Cited on pages 28 and 30.

IEEE, "IEEE draft standard for local and metropolitan area networks - timing and synchronization for time-sensitive applications," *IEEE P802.1AS-Rev/D8.0 January, 2019*, pp. 1–446, 2019. [Online]. Available: https://1.ieee802.org/tsn/802-1as-rev/ Cited on page 30.

IEEE, "Ieee standard for local and metropolitan area networks– audio video bridging (avb) systems– corrigendum 1: Technical and editorial corrections," *IEEE Std 802.1BA-2011/Cor 1-2016 (Corrigendum to IEEE Std 802.1BA-2011)*, pp. 1–13, 2016. [Online]. Available: https://doi.org/10.1109/IEEESTD.2016.7520635 Cited on page 46.

IEEE, "Draft standard for local and metropolitan area networks — time-sensitive networking profile for automotive in-vehicle ethernet communications," *IEEE P802.1DG/D1.4)*, pp. 1–137, 2021. [Online]. Available: https://www.ieee802.org/1/files/private/dg-drafts/d1/802-1DG-d1-4.pdf Cited on pages 29 and 34.

IEEE, "Ieee standard for local and metropolitan area networks–bridges and bridged networks," *IEEE Std 802.1Q-2014 (Revision of IEEE Std 802.1Q-2011)*, pp. 1–1832, 2014. [Online]. Available: https://doi.org/10.1109/IEEESTD.2014.6991462 Cited on pages 8, 24, 31, and 40.

IEEE, "IEEE standard for local and metropolitan area networks–virtual bridged local area networks amendment 14: Stream reservation protocol (srp)," *IEEE Std 802.1Qat-2010 (Revision of IEEE Std 802.1Q-2005)*, pp. 1–119, 2010. [Online]. Available: https://doi.org/10.1109/IEEESTD.2010.5594972 Cited on pages 28, 30, and 31.

IEEE, "IEEE standard for local and metropolitan area networks - virtual bridged local area networks amendment 12: Forwarding and queuing enhancements for time-sensitive streams," *IEEE Std 802.1Qav-2009 (Amendment to IEEE Std 802.1Q-2005)*, pp. C1–72, 2010. [Online]. Available: https://doi.org/10.1109/IEEESTD.2009.5375704 Cited on pages 28 and 34.

IEEE, "Standard for local and metropolitan area networks – bridges and bridged networks – amendment 26: Frame preemption," *IEEE Std 802.1Qbu-2016 (Amendment to IEEE Std 802.1Q-2014)*, pp. 1–52, 2016. [Online]. Available: https://doi.org/10.1109/IEEESTD.2016.7553415 Cited on pages 4, 36, and 37.

IEEE, "Standard for local and metropolitan area networks – bridges and bridged networks - amendment 25: Enhancements for scheduled traffic," *IEEE Std 802.1Qbv-2015 (Amendment to IEEE Std 802.1Q-2014 as amended by IEEE Std 802.1Qca-2015, IEEE Std 802.1Qcd-2015, and IEEE Std 802.1Q-2014/Cor 1-2015)*, pp. 1–57, 2016. [Online]. Available: https://doi.org/10.1109/IEEESTD.2016.8613095 Cited on pages 5, 32, and 49.

IEEE, "IEEE standard for local and metropolitan area networks–bridges and bridged networks–amendment 29: Cyclic queuing and forwarding," *IEEE 802.1Qch-2017 (Amendment to IEEE Std 802.1Q-2014 as amended by IEEE Std 802.1Qca-2015, IEEE Std 802.1Qcd(TM)-2015, IEEE Std 802.1Q-2014/Cor 1-2015, IEEE Std 802.1Qbv-2015, IEEE Std 802.1Qbu-2016, IEEE Std 802.1Qbz-2016, and IEEE Std 802.1Qci-2017)*, pp. 1–30, 2017. [Online]. Available: https://doi.org/10.1109/IEEESTD.2017.7961303

IEEE, "IEEE standard for local and metropolitan area networks–bridges and bridged networks–amendment 28: Per-stream filtering and policing," *IEEE Std 802.1Qci-2017 (Amendment to IEEE Std 802.1Q-2014 as amended by IEEE Std 802.1Qca-2015, IEEE Std 802.1Qcd-2015, IEEE Std 802.1Q-2014/Cor 1-2015, IEEE Std 802.1Qbv-2015, IEEE Std 802.1Qbu-2016, and IEEE Std 802.1Qbz-2016)*, pp. 1–65, 2017. [Online]. Available: https://doi.org/10.1109/IEEESTD.2017.8064221 Cited on pages 32 and 35.

IEEE, "Bridges and Bridged Networks—Amendment: Asynchronous Traffic Shaping," 2019. [Online]. Available: www.ieee802.org/1/files/private/cr-drafts/d1/802-1Qcr-d1-1.pdf Cited on page 36.

IEEE, "Ieee standard for local and metropolitan area networks–media access control (mac) bridges and virtual bridged local area networks–amendment 20: Shortest path bridging," pp. 1–340, 2012. [Online]. Available: https://doi.org/10.1109/IEEESTD.2012.6231597

IEEE, "Ieee standard for ethernet amendment 5: Specification and management parameters for interspersing express traffic," *IEEE Std 802.3br-2016 (Amendment to IEEE Std 802.3-2015 as amended by IEEE St802.3bw-2015, IEEE Std 802.3by-2016, IEEE Std 802.3bq-2016, and IEEE Std 802.3bp-2016)*, pp. 1–58, 2016. [Online]. Available: https://doi.org/10.1109/IEEESTD.2016.7900321 Cited on pages 4, 36, 37, 40, 42, 53, and 54.

IEEE, "IEEE draft standard for local and metropolitan area networks–media access control (mac) bridges and virtual bridged local area networks amendment: Stream reservation protocol (srp) enhancements and performance improvements," *IEEE P802.1Qcc/D2.2, March 2018*, pp. 1–214, 2018. [Online]. Available: https://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=8315187

IEEE, "IEEE standard for local area network mac (media access control) bridges," *ANSI/IEEE Std 802.1D, 1998 Edition*, pp. 1–373, 1998. [Online]. Available: https://doi.org/10.1109/IEEESTD.1998.95619

IEEE, "Standard for local and metropolitan area networks–frame replication and elimination for reliability," *IEEE Std 802.1CB-2017*, pp. 1–102, 2017. [Online]. Available: https://doi.org/10.1109/IEEESTD.2017.8091139 Cited on page 31.

IEEE, "Ieee standard for local and metropolitan area networks— bridges and bridged networks - amendment 24: Path control and reservation," *IEEE Std 802.1Qca-2015 (Amendment to IEEE Std 802.1Q-2014 as amended by IEEE Std 802.1Qcd-2015 and IEEE Std 802.1Q-2014/Cor 1-2015)*, pp. 1–120, 2016. [Online]. Available: https://doi.org/10.1109/IEEESTD.2016.7434544 Cited on pages 31, 50, and 119.

IEEE-TSN, "Time-Sensitive Networking Task Group," 2012. [Online]. Available: http://www.IEEE802.org/1/pages/tsn.html Cited on pages 4, 10, and 29.

D. Jansen and H. Buttner, "Real-time ethernet: the ethercat solution," *Computing and Control Engineering*, vol. 15, no. 1, pp. 16–21, 2004. [Online]. Available: https://doi.org/10.1049/cce:20040104 Cited on page 2.

W.-K. Jia, G.-H. Liu, and Y.-C. Chen, "Performance evaluation of ieee 802.1qbu: Experimental and simulation results," in *38th IEEE Conference on Local Computer Networks*, 2013, pp. 659–662. [Online]. Available: https://doi.org/10.1109/LCN.2013.6761304 Cited on pages 2, 5, 45, and 47.

M. D. Johas Teener and G. M. Garner, "Overview and timing performance of ieee 802.1as," in *IEEE International Symposium on Precision Clock Synchronization for Measurement, Control and Communication*, 2008, pp. 49–53. [Online]. Available: https://doi.org/10.1109/ISPCS.2008.4659212 Cited on page 30.

D. Kaljun and J. Žerovnik, "On local search based heuristics for optimization problems," *Croatian Operational Research Review*, vol. 5, no. 2, pp. 317–327, 2014. [Online]. Available: https://hrcak.srce.hr/ojs/index.php/crorr/article/view/2755

S. Kamal, S. Al Mubarak, B. Scodova, P. Naik, P. Flichy, G. Coffin *et al.*, "IT and OT convergence-opportunities and challenges," in *SPE Intelligent Energy International Conference and Exhibition*, 2016. [Online]. Available: https://doi.org/10.2118/181087-MS

S. Kehrer, O. Kleineberg, and D. Heffernan, "A comparison of fault-tolerance concepts for ieee 802.1 time sensitive networks (tsn)," in *IEEE Emerging Technology and Factory Automation (ETFA)*, 2014, pp. 1–8. [Online]. Available: https://doi.org/10.1109/ETFA.2014.7005200

J. Kim, B. Y. Lee, and J. Park, "Preemptive switched ethernet for real-time process control system," in *11th IEEE International Conference on Industrial Informatics (INDIN)*, 2013, pp. 171–176. [Online]. Available: https://doi.org/10.1109/INDIN.2013.6622877 Cited on pages 45 and 47.

Y. Kim, "Very low latency packet delivery requirements and problem statements," 2011. [Online]. Available: https://www.ieee802.org/1/files/public/docs2011/new-avb-kim-very-low-latency-packet-delivery-problem-statements-1111-v01.pdf

M. Knezic, M. Kovacevic, and Z. Ivanovic, "Implementation aspects of multi-level frame preemption in tsn," in *25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, vol. 1, 2020, pp. 1127–1130. [Online]. Available: https://doi.org/10.1109/ETFA46521.2020.9211914 Cited on pages 48 and 135.

H. Kopetz, A. Ademaj, P. Grillinger, and K. Steinhammer, "The time-triggered ethernet (tte) design," in *8th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'05)*, 2005, pp. 22–33. [Online]. Available: https://doi.org/10.1109/ISORC.2005.56 Cited on page 28.

J.-Y. Le Boudec and P. Thiran, *Network calculus: a theory of deterministic queuing systems for the internet*. Springer Berlin, Heidelberg, 2001, vol. 2050. [Online]. Available: https://doi.org/10.1007/3-540-45318-0

H. Lee, J. Lee, C. Park, and S. Park, "Time-aware preemption to enhance the performance of audio/video bridging (avb) in ieee 802.1 tsn," in *2016 First IEEE International Conference on Computer Communication and the Internet (ICCCI)*, 2016, pp. 80–84. [Online]. Available: https://doi.org/10.1109/CCI.2016.7778882

S. Lee, H. Baek, H. Woo, K. G. Shin, and J. Lee, "Ml for rt: Priority assignment using machine learning," in *27th Real-Time and Embedded Technology and Applications Symposium (RTAS)*. Online: IEEE, 2021, pp. 118–130. [Online]. Available: https://doi.org/10.1109/RTAS52030.2021.00018 Cited on pages 49, 50, 88, and 110.

X. Li, O. Cros, and L. George, "The trajectory approach for afdx fifo networks revisited and corrected," in *IEEE 20th International Conference on Embedded and Real-Time Computing Systems and Applications*, 2014, pp. 1–10. [Online]. Available: https://doi.org/10.1109/RTCSA.2014.6910523 Cited on page 47.

L. Lo Bello and W. Steiner, "A perspective on IEEE time-sensitive networking for industrial communication and automation systems," *Proceedings of the IEEE*, vol. 107, no. 6, pp. 1094–1120, 2019. [Online]. Available: https://doi.org/10.1109/JPROC.2019.2905334 Cited on pages 4, 5, 6, 14, and 44.

L. Lo Bello, M. Ashjaei, G. Patti, and M. Behnam, "Schedulability analysis of time-sensitive networks with scheduled traffic and preemption support," *Journal of Parallel and Distributed Computing*, vol. 144, pp. 153–171, 2020. [Online]. Available: https://doi.org/10.1016/j.jpdc.2020.06.001 Cited on page 93.

L. Lo Bello, M. Ashjaei, G. Patti, and M. Behnam, "Schedulability analysis of time-sensitive networks with scheduled traffic and preemption support," *Journal of Parallel and Distributed Computing*, vol. 144, pp. 153–171, 2020. [Online]. Available: https://doi.org/10.1016/j.jpdc.2020.06.001 Cited on page 48.

A. Lodi, S. Martello, and D. Vigo, "Recent advances on two-dimensional bin packing problems," *Discrete Applied Mathematics*, vol. 123, no. 1, pp. 379–396, 2002. [Online]. Available: https://doi.org/10.1016/S0166-218X(01)00347-X Cited on page 88.

J. Lund, "Time Sensitive Networking - Applications and Readiness," 2017. [Online]. Available: https://www.iiconsortium.org/berlin-forum-2017/Lund-Time-Sensitive-Networking-Applications-and-Readiness.pdf

R. Mahfouzi, A. Aminifar, S. Samii, A. Rezine, P. Eles, and Z. Peng, "Stability-aware integrated routing and scheduling for control applications in ethernet networks," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2018, pp. 682–687. [Online]. Available: https://doi.org/10.23919/DATE.2018.8342096

T. L. Mai and N. Navet, "Deep learning to predict the feasibility of priority-based ethernet network configurations," *Transactions on Cyber-Physical Systems (TCPS)*, vol. 5, no. 4, pp. 1–26, 2021. [Online]. Available: https://doi.org/10.1145/3468890 Cited on page 50.

T. L. Mai, N. Navet, and J. Migge, "On the use of supervised machine learning for assessing schedulability: Application to ethernet tsn," in *27th International Conference on Real-Time Networks and Systems*, ser. RTNS '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 143–153. [Online]. Available: https://doi.org/10.1145/3356401.3356409 Cited on page 50.

T. L. Mai, N. Navet, and J. Migge, "A hybrid machine learning and schedulability analysis method for the verification of tsn networks," in *15th IEEE International Workshop on Factory Communication Systems (WFCS)*, 2019, pp. 1–8. [Online]. Available: https://doi.org/10.1109/WFCS.2019.8757948 Cited on page 50.

N. Malcolm and W. Zhao, "Advances in hard real-time communication with local area networks," in *[1992] Proceedings 17th Conference on Local Computer Networks*, 1992, pp. 548–557. [Online]. Available: https://doi.org/10.1109/LCN.1992.228144 Cited on page 17.

S. Martin and P. Minet, "Worst case end-to-end response times of flows scheduled with fp/fifo," in *International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies (ICNICONSMCL'06)*, 2006, pp. 54–54. [Online]. Available: https://doi.org/10.1109/ICNICONSMCL.2006.231 Cited on page 47.

S. Martin and P. Minet, "Schedulability analysis of flows scheduled with fifo: application to the expedited forwarding class," in *20th IEEE International Parallel & Distributed Processing Symposium*, 2006, pp. 8 pp.–. [Online]. Available: https://doi.org/10.1109/IPDPS.2006.1639424

G. I. Mary, Z. C. Alex, and L. Jenkins, "Response time analysis of messages in Controller Area Network: a review," *Journal of Computer Networks and Communications*, vol. 2013, 2013. [Online]. Available: https://doi.org/10.1155/2013/148015

C. Mauclair, M. Gutiérrez, J. Migge, and N. Navet, "Do we really need tsn in next-generation helicopters? insights from a case-study," in *IEEE/AIAA 40th Digital Avionics Systems Conference (DASC)*, 2021, pp. 1–7. [Online]. Available: https://doi.org/10.1109/DASC52595.2021.9594349

D. Maxim and Y.-Q. Song, "Delay analysis of avb traffic in time-sensitive networks (tsn)," in *25th International Conference on Real-Time Networks and Systems*. New York, NY, USA: Association for Computing Machinery, 2017. [Online]. Available: https://doi.org/10.1145/3139258.3139283 Cited on page 48.

P. Meyer, T. Steinbach, F. Korf, and T. C. Schmidt, "Extending IEEE 802.1 avb with time-triggered scheduling: A simulation study of the coexistence of synchronous and asynchronous traffic," in *IEEE Vehicular Networking Conference*, 2013, pp. 47–54. [Online]. Available: https://doi.org/10.1109/VNC.2013.6737589 Cited on page 46.

J. Migge, J. Villanueva, N. Navet, and M. Boyer, "Insights on the performance and configuration of avb and tsn in automotive ethernet networks," *Proc. Embedded Real-Time Software and Systems (ERTS 2018)*, 2018. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01746132 Cited on page 106.

A. Nasrallah, A. S. Thyagaturu, Z. Alharbi, C. Wang, X. Shao, M. Reisslein, and H. ElBakoury, "Ultra-low latency (ull) networks: The IEEE tsn and ietf detnet standards and related 5g ull research," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 1, pp. 88–145, 2019. [Online]. Available: https://doi.org/10.1109/COMST.2018.2869350 Cited on pages 4, 24, 30, 31, 32, and 50.

A. Nasrallah, A. S. Thyagaturu, Z. Alharbi, C. Wang, X. Shao, M. Reisslein, and H. Elbakoury, "Performance comparison of IEEE 802.1 tsn time aware shaper (tas) and asynchronous traffic shaper (ats)," *IEEE Access*, vol. 7, pp. 44 165–44 181, 2019. [Online]. Available: https://doi.org/10.1109/ACCESS.2019.2908613 Cited on pages 46 and 59.

N. Navet, T. L. Mai, and J. Migge, "Using machine learning to speed up the design space exploration of ethernet tsn networks," University of Luxembourg, Tech. Rep., 2019. Cited on page 50.

N. G. Nayak, F. Dürr, and K. Rothermel, "Routing algorithms for ieee802.1qbv networks," *ACM SIGBED Reivew*, vol. 15, no. 3, 2018. [Online]. Available: https://doi.org/10.1145/3267419.3267421 Cited on pages 13, 28, 51, and 119.

M. A. Ojewale, P. M. Yomsi, and B. Nikolić, "Multi-Level Preemption in TSN: Feasibility and Requirements Analysis," in *IEEE 23rd International Symposium on Real-Time Distributed Computing (ISORC)*, May 2020, pp. 47–55. [Online]. Available: https://doi.org/10.1109/ISORC49007.2020.00017 Cited on page 49.

M. A. Ojewale and P. M. Yomsi, "Routing heuristics for load-balanced transmission in tsn-based networks," *SIGBED Rev.*, vol. 16, no. 4, p. 20–25, Jan. 2020. [Online]. Available: https://doi.org/10.1145/3378408.3378411 Cited on page 31.

M. A. Ojewale, P. Meumeu Yomsi, and G. Nelissen, "On multi-level preemption in ethernet," in *Proceedings of the Work-in-Progress Session (ECRTS)*, 2018, pp. 16–18. Cited on page 78.

M. A. Ojewale, P. M. Yomsi, and B. Nikolić, "Worst-case traversal time analysis of tsn with multi-level preemption," *Journal of Systems Architecture*, vol. 116, p. 102079, 2021. [Online]. Available: https://doi.org/10.1016/j.sysarc.2021.102079 Cited on page 93.

OMNET++, "Discrete Event Simulator." [Online]. Available: https://omnetpp.org/ Cited on page 12.

A. Oonsivilai, W. Srisuruk, B. Marungsri, and T. Kulworawanichpong, "Tabu search approach to solve routing issues in communication networks," *International Journal of Electronics and Communication Engineering*, vol. 3, no. 5, pp. 1211 – 1214, 2009. [Online]. Available: https://publications.waset.org/vol/29

F. OPC, " Open platform communications unified architecture." [Online]. Available: https://opcfoundation.org/about/opc-technologies/opc-ua/

M. Pahlevan and R. Obermaisser, "Evaluation of time-triggered traffic in time-sensitive networks using the opnet simulation framework," in *26th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, 2018, pp. 283–287. [Online]. Available: https://doi.org/10.1109/PDP2018.2018.00048

M. K. Pakhira, "A linear time-complexity k-means algorithm using cluster shifting," in *International Conference on Computational Intelligence and Communication Networks*. Bhopal, India: IEEE, 2014, pp. 1047–1051. [Online]. Available: https://doi.org/10.1109/CICN.2014.220 Cited on page 112.

D. Pannell, "IEEE TSN Standards Overview & Update," 2015. [Online]. Available: https://standards.ieee.org/content/dam/ieee-standards/standards/web/documents/presentations/d2-08_avnu_ieee-802.1-tsn_standards_overview_and_update_v2.pdf

D. Park, J. Lee, C. Park, and S. Park, "New automatic de-registration method utilizing a timer in the ieee802.1 tsn," in *2016 First IEEE International Conference on Computer Communication and the Internet (ICCCI)*, 2016, pp. 47–51. [Online]. Available: https://doi.org/10.1109/CCI.2016.7778875

T. Park and K. G. Shin, "Optimal priority assignment for scheduling mixed can and can-fd frames," in *Real-Time and Embedded Technology and Applications Symposium (RTAS)*. Montreal, Canada: IEEE, 2019, pp. 192–203. [Online]. Available: https://doi.org/10.1109/RTAS.2019.00024 Cited on pages 49 and 88.

T. Park, S. Samii, and K. G. Shin, "Design optimization of frame preemption in real-time switched ethernet," in *2019 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2019, pp. 420–425. [Online]. Available: https://doi.org/10.23919/DATE.2019.8714953 Cited on page 50.

J. Parmar, "Automotive Electronics – Electronic control unit," https://electronicsforu.com/market-verticals/automotive-electronic-control-unit, accessed: 2019-11-06.

J. Pfrommer, A. Ebner, S. Ravikumar, and B. Karunakaran, "Open source opc ua pubsub over tsn for realtime industrial communication," in *International Conference on Emerging Technologies and Factory Automation*, 2018, pp. 1087–1090. [Online]. Available: https://doi.org/10.1109/ETFA.2018.8502479

P. Pop, M. L. Raagaard, S. S. Craciunas, and W. Steiner, "Design Optimisation of Cyber-physical Distributed Systems Using IEEE Time-sensitive Networks," *IET Cyber-Physical Systems: Theory & Applications*, vol. 1, pp. 86–94, 2016. [Online]. Available: https://doi.org/10.1049/iet-cps.2016.0021 Cited on page 50.

P. Pop, M. L. Raagaard, M. Gutierrez, and W. Steiner, "Enabling fog computing for industrial automation through time-sensitive networking (tsn)," *IEEE Communications Standards Magazine*, vol. 2, no. 2, pp. 55–61, 2018. [Online]. Available: https://doi.org/10.1109/MCOMSTD.2018.1700057

T. Pop, P. Pop, P. Eles, Z. Peng, and A. Andrei, "Timing analysis of the flexray communication protocol," in *18th Euromicro Conference on Real-Time Systems (ECRTS'06)*, 2006, pp. 11 pp.–216. [Online]. Available: https://doi.org/10.1109/ECRTS.2006.31 Cited on pages 2 and 23.

A. Pruski, M. A. Ojewale, V. Gavrilut, P. M. Yomsi, M. S. Berger, and L. Almeida, "Implementation cost comparison of tsn traffic control mechanisms," in *26th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2021, pp. 01–08. [Online]. Available: https://doi.org/10.1109/ETFA45728.2021.9613463 Cited on pages 14, 60, and 63.

M. L. Raagaard, P. Pop, M. Gutiérrez, and W. Steiner, "Runtime reconfiguration of time-sensitive networking (tsn) schedules for fog computing," in *2017 IEEE Fog World Congress (FWC)*, 2017, pp. 1–6. [Online]. Available: https://doi.org/10.1109/FWC.2017.8368523

F. Reimann, S. Graf, F. Streit, M. Glaß, and J. Teich, "Timing analysis of ethernet avb-based automotive e/e architectures," in *IEEE 18th Conference on Emerging Technologies & Factory Automation (ETFA)*, 2013, pp. 1–8. [Online]. Available: https://doi.org/10.1109/ETFA.2013.6648024 Cited on page 47.

N. Reusch, L. Zhao, S. S. Craciunas, and P. Pop, "Window-based schedule synthesis for industrial IEEE 802.1qbv tsn networks," in *16th IEEE International Conference on Factory Communication Systems (WFCS)*, 2020, pp. 1–4. [Online]. Available: https://doi.org/10.1109/WFCS47810.2020.9114414 Cited on page 49.

P. Richards, "A can physical layer discussion," *Microchip Technology Inc*, 2002. Cited on pages 21 and 22.

J. Rox and R. Ernst, "Formal Timing Analysis of Full Duplex Switched Based Ethernet Network Architectures," in *SAE 2010 World Congress & Exhibition*, 2010, p. 12. [Online]. Available: https://doi.org/10.4271/2010-01-0455 Cited on pages 48 and 76.

R. Salazar, T. Godfrey, N. Finn, C. Powell, B. Rolfe, and M. Seewald, "Utility applications of time sensitive networking white paper," *Utility Applications of Time Sensitive Networking White Paper*, pp. 1–19, 2019. [Online]. Available: https://www.ieee802.org/24/Utility%20Applications%20of%20Time%20Sensitive%20Networking_white%20paper_final%20review.pdf

E. Schemm, "Sercos to link with ethernet for its third generation," *Computing and Control Engineering*, vol. 15, no. 2, pp. 30–33, 2004. [Online]. Available: https://doi.org/10.1049/cce:20040205 Cited on page 2.

R. Serna Oliver, S. S. Craciunas, and G. Stöger, "Analysis of deterministic ethernet scheduling for the industrial internet of things," in *IEEE 19th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, 2014, pp. 320–324. [Online]. Available: https://doi.org/10.1109/CAMAD.2014.7033258

R. Serna Oliver, S. S. Craciunas, and W. Steiner, "IEEE 802.1qbv gate control list synthesis using array theory encoding," in *IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, 2018, pp. 13–24. [Online]. Available: https://doi.org/10.1109/RTAS.2018.00008 Cited on page 49.

J. F. Shoch, "An introduction to the ethernet specification," *SIGCOMM Comput. Commun. Rev.*, vol. 11, no. 3, p. 17–19, jul 1981. [Online]. Available: https://doi.org/10.1145/1015591.1015593 Cited on pages 2 and 23.

L. Silva, P. Pedreiras, P. Fonseca, and L. Almeida, "On the adequacy of sdn and tsn for industry 4.0," in *IEEE 22nd International Symposium on Real-Time Distributed Computing (ISORC)*, 2019, pp. 43–51. [Online]. Available: https://doi.org/10.1109/ISORC.2019.00017

C. Simon, M. Máté, M. Maliosz, and N. Bella, "Ethernet with time sensitive networking tools for industrial networks," *Infocommunications Journal*, vol. 9, no. 2, pp. 6–14, 2017. [Online]. Available: https://www.infocommunications.hu/documents/169298/3776623/InfocomJ_2017_2_2_Simon.pdf Cited on page 45.

S. Singh, "Routing Algorithms for Time Sensitive Networks," Master's thesis, University of Stuttgart, 2017. [Online]. Available: https://elib.uni-stuttgart.de/handle/11682/9499 Cited on pages 13, 31, 51, 119, and 120.

F. Smirnov, M. Glaß, F. Reimann, and J. Teich, "Formal timing analysis of non-scheduled traffic in automotive scheduled tsn networks," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*, 2017, pp. 1643–1646. [Online]. Available: https://doi.org/10.23919/DATE.2017.7927256

J. Specht and S. Samii, "Urgency-based scheduler for time-sensitive switched ethernet networks," in *28th Euromicro Conference on Real-Time Systems (ECRTS)*, 2016, pp. 75–85. [Online]. Available: https://doi.org/10.1109/ECRTS.2016.27 Cited on page 36.

J. Specht and S. Samii, "Synthesis of queue and priority assignment for asynchronous traffic shaping in switched ethernet," in *IEEE Real-Time Systems Symposium (RTSS)*, 2017, pp. 178–187. [Online]. Available: https://doi.org/10.1109/RTSS.2017.00024 Cited on page 49.

J. W. Specks and A. Rajnák, "Lin-protocol, development tools, and software interfaces for local interconnect networks in vehicles," *VDI-Berichte*, pp. 227–250, 2000. Cited on pages 2 and 23.

R. P. Stanley, "Enumerative combinatorics volume 1 second edition," *Cambridge studies in advanced mathematics*, 2011. [Online]. Available: https://www.ms.uky.edu/~sohum/putnam/enu_comb_stanley.pdf Cited on page 96.

W. Steiner, "An evaluation of smt-based schedule synthesis for time-triggered multi-hop networks," in *31st IEEE Real-Time Systems Symposium*, 2010, pp. 375–384. [Online]. Available: https://doi.org/10.1109/RTSS.2010.25

W. Steiner, G. Bauer, B. Hall, M. Paulitsch, and S. Varadarajan, "Ttethernet dataflow concept," in *8th IEEE International Symposium on Network Computing and Applications*, 2009, pp. 319–322. [Online]. Available: https://doi.org/10.1109/NCA.2009.28 Cited on pages 2, 28, and 49.

C. P. Szydlowski, "Can specification 2.0: Protocol and implementations," in *Future Transportation Technology Conference & Exposition*. SAE International, aug 1992. [Online]. Available: https://doi.org/10.4271/921603 Cited on pages 2 and 22.

D. Tamas-Selicean, P. Pop, and W. Steiner, "Synthesis of communication schedules for ttethernet-based mixed-criticality systems," in *8th IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*, ser. CODES+ISSS '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 473–482. [Online]. Available: https://doi.org/10.1145/2380445.2380518 Cited on page 49.

D. Tămaş-Selicean, P. Pop, and W. Steiner, "Design optimization of TTEthernet-based distributed real-time systems," *Real-Time Systems*, vol. 51, no. 1, pp. 1–35, 2015. [Online]. Available: https://doi.org/10.1007/s11241-014-9214-8

M. J. Teener, "Back to the future: using tas and preemption for deterministic distributed delays," in *IEEE 802.1 AVB TG Meeting*, 2012. [Online]. Available: https://www.ieee802.org/1/files/public/docs2012/new-avb-mjt-back-to-the-future-1112-v01.pdf

S. Thangamuthu, N. Concer, P. J. L. Cuijpers, and J. J. Lukkien, "Analysis of ethernet-switch traffic shapers for in-vehicle networking applications," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2015, pp. 55–60. [Online]. Available: https://doi.org/10.7873/DATE.2015.0045 Cited on pages 46 and 48.

D. Thiele and R. Ernst, "Formal worst-case timing analysis of ethernet tsn's burst-limiting shaper," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 2016, pp. 187–192. [Online]. Available: https://ieeexplore.ieee.org/document/7459302 Cited on pages 34, 48, and 76.

D. Thiele and R. Ernst, "Formal worst-case performance analysis of time-sensitive ethernet with frame preemption," in *IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2016, pp. 1–9. [Online]. Available: https://doi.org/10.1109/ETFA.2016.7733740 Cited on pages 2, 5, 14, 44, 45, 47, 48, 59, 74, 76, 77, 78, 80, 81, and 83.

D. Thiele, P. Axer, R. Ernst, and J. R. Seyler, "Improving formal timing analysis of switched ethernet by exploiting traffic stream correlations," in *International Conference on Hardware/Software Codesign and System Synthesis*, ser. CODES '14. New York, NY, USA: Association for Computing Machinery, 2014. [Online]. Available: https://doi.org/10.1145/2656075.2656090 Cited on pages 48 and 83.

D. Thiele, P. Axer, and R. Ernst, "Improving formal timing analysis of switched ethernet by exploiting fifo scheduling," in *52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2015, pp. 1–6. [Online]. Available: https://doi.org/10.1145/2744769.2744854 Cited on pages 48 and 76.

D. Thiele, R. Ernst, and J. Diemer, "Formal worst-case timing analysis of ethernet tsn's time-aware and peristaltic shapers," in *IEEE Vehicular Networking Conference (VNC)*, 2015, pp. 251–258. [Online]. Available: https://doi.org/10.1109/VNC.2015.7385584 Cited on pages 46 and 48.

L. Thiele, S. Chakraborty, and M. Naedele, "Real-time calculus for scheduling hard real-time systems," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 4, 2000, pp. 101–104 vol.4. [Online]. Available: https://doi.org/10.1109/ISCAS.2000.858698

B. Wang and J. Hou, "Multicast routing and its qos extension: problems, algorithms, and protocols," *IEEE Network*, vol. 14, no. 1, pp. 22–36, 2000. [Online]. Available: https://doi.org/10.1109/65.819168 Cited on page 50.

Z. Wang and J. Crowcroft, "Quality-of-service routing for supporting multimedia applications," *IEEE Journal on Selected Areas in Communications*, vol. 14, no. 7, pp. 1228–1234, 1996. [Online]. Available: https://doi.org/10.1109/49.536364

L. Winkel, "Real-time ethernet in iec 61784-2 and iec 61158 series," in *4th IEEE International Conference on Industrial Informatics*, 2006, pp. 246–250. [Online]. Available: https://doi.org/10.1109/INDIN.2006.275788 Cited on page 27.

Xilinx, "UltraScale Architecture and Product Data Sheet: Overview (DS890)," 2020. Cited on page 61.

Xilinx, "UltraScale Architecture Configurable Logic Block User Guide (UG574)," 2017. Cited on page 61.

X. Zeng and D. Song, "The research on end-to-end delay calculation method for real-time network afdx," in *International Conference on Computational Intelligence and Software Engineering*, 2009, pp. 1–4. [Online]. Available: https://doi.org/10.1109/CISE.2009.5363642

L. Zhao, F. He, E. Li, and H. Xiong, "Improving worst-case delay analysis for traffic of additional stream reservation class in ethernet-avb network," *Sensors*, vol. 18, no. 11, 2018. [Online]. Available: https://www.mdpi.com/1424-8220/18/11/3849 Cited on page 34.

L. Zhao, P. Pop, Q. Li, J. Chen, and H. Xiong, "Timing analysis of rate-constrained traffic in ttethernet using network calculus," *Real-Time Systems*, vol. 53, no. 2, pp. 254–287, 2017. [Online]. Available: https://doi.org/10.1007/s11241-016-9265-0 Cited on pages xiv and 116.

L. Zhao, P. Pop, and S. S. Craciunas, "Worst-case latency analysis for IEEE 802.1qbv time sensitive networks using network calculus," *IEEE Access*, vol. 6, pp. 41 803–41 815, 2018. [Online]. Available: https://doi.org/10.1109/ACCESS.2018.2858767 Cited on page 47.

L. Zhao, P. Pop, Z. Zheng, H. Daigmorte, and M. Boyer, "Latency analysis of multiple classes of avb traffic in tsn with standard credit behavior using network calculus," *IEEE Transactions on Industrial Electronics*, vol. 68, no. 10, pp. 10 291–10 302, 2021. [Online]. Available: https://doi.org/10.1109/TIE.2020.3021638 Cited on page 47.

Z. Zhou, Y. Yan, S. Ruepp, and M. Berger, "Analysis and implementation of packet preemption for time sensitive networks," in *IEEE 18th International Conference on High Performance Switching and Routing (HPSR)*, 2017, pp. 1–6. [Online]. Available: https://doi.org/10.1109/HPSR.2017.7968677 Cited on pages 45 and 47.

Z. Zhou, M. S. Berger, and Y. Yan, "Mapping tsn traffic scheduling and shaping to fpga-based architecture," *IEEE Access*, vol. 8, pp. 221 503–221 512, 2020. [Online]. Available: https://doi.org/10.1109/ACCESS.2020.3043887

Z. Zhou, J. Lee, M. S. Berger, S. Park, and Y. Yan, "Simulating tsn traffic scheduling and shaping for future automotive ethernet," *Journal of Communications and Networks*, vol. 23, no. 1, pp. 53–62, 2021. [Online]. Available: https://doi.org/10.23919/JCN.2021.000001