THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Institut Supérieur de l'Aéronautique et de l'Espace

Présentée et soutenue par :
Anaïs FINZI

le lundi 11 juin 2018

Titre :

Specification and analysis of an extended AFDX with TSN/BLS shapers for mixed-criticality avionics applications

Spécification et analyse d'un AFDX étendu avec TSN/BLS pour des applications avioniques de criticités mixtes

École doctorale et discipline ou spécialité :

ED MITT : Réseaux, télécom, système et architecture

Unité de recherche :

Équipe d'accueil ISAE-ONERA MOIS

Directeur(s) de Thèse :

M. Emmanuel LOCHIN (directeur de thèse)
Mme Ahlem MIFDAOUI (co-directrice de thèse)

Jury :

M. Jean-Yves LE BOUDEC, Professeur EPFL - Président
M. Fabrice FRANCES, Professeur Associé ISAE-SUPAERO
M. Laurent GEORGE, Professeur ESIEE Paris - Rapporteur
M. Emmanuel LOCHIN, Professeur ISAE-SUPAERO - Directeur de thèse
Mme Ahlem MIFDAOUI, Professeure ISAE-SUPAERO - Co-directrice de thèse
M. Ye-Qiong SONG, Professeur Université de Lorraine - Rapporteur

RÉSUMÉ

L'augmentation du nombre de systèmes interconnectés et l'expansion des données échangées dans les réseaux avioniques ont contribué à la complexification des architectures de communication. Pour gérer cette évolution, une nouvelle solution basée sur un réseau coeur haut débit, e.g., l'AFDX (Avionics Full DupleX), a été implémentée sur l'A380. Cependant, il reste des réseaux bas débit, e.g, CAN ou A429, utilisés pour certaines fonctions spécifiques. Cette architecture réduit le délai de développement, mais en contrepartie, elle conduit à de l'hétérogénéité et à de nouveaux challenges pour garantir les contraintes temps-réel.

Pour résoudre ces challenges, une architecture homogène basée sur l'AFDX pourrait apporter de grands avantages, tels que une facilité de l'installation et maintenance, et une réduction de poids et coûts. Cette architecture homogène doit supporter des applications de criticités mixtes, où coexistent les trafics critiques (SCT), *Best-effort* (BE) et le trafic AFDX actuel (RC).

Pour atteindre ce but, nous commençons par évaluer les avantages et les inconvénients des solutions existantes par rapport aux contraintes avioniques.

Cela nous conduit à sélectionner le *Burst Limiting Shaper* (BLS) (proposé par le groupe *IEEE Time Sensitive Networking* (TSN)) allié à un ordonnanceur *Static Priority* non-preemptif.

Ainsi, nous identifions quatre contributions principales dans cette thèse. Tout d'abord, nous spécifions un AFDX étendu avec le TSN/BLS. Une analyse préliminaire basée sur de la simulation a donné des résultats encourageants pour poursuivre sur cette voie.

En second, nous détaillons une analyse temporelle de l'AFDX étendu, grâce au Network Calculus, pour calculer des bornes maximales des délais pire cas des différents types de trafic, pour prouver le déterminisme du réseau et le respect des contraintes temporelles. Une analyse de performance préliminaire montre l'efficacité de la solution à améliorer les délais de RC, tout en garantissant les contraintes du SCT. Cependant, cette analyse a aussi montré certaines limitations du modèle en termes de pessimisme.

Notre troisième contribution est par conséquent la réduction de ce pessimisme, grâce à une seconde modélisation de l'AFDX étendu, et à une méthode de paramétrage des variables système. Cette méthode permet d'améliorer les performances de RC, tout en garantissant les contraintes temporelles du SCT et RC.

Finalement, nous validons notre proposition à travers des études de cas avioniques réalistes pour vérifier son efficacité. Les résultats montrent une forte amélioration des délais de RC ainsi que de l'ordonnançabilité de SCT et RC, en comparaison à l'AFDX actuel et au Deficit Round Robin.

ABSTRACT

The growing number of interconnected end-systems and the expansion of exchanged data in avionics have led to an increase in complexity of the communication architecture. To cope with this trend, a first communication solution based on a high rate backbone network, i.e., the AFDX (Avionics Full Duplex Switched Ethernet), has been implemented by Airbus in the A380. Moreover, some low rate data buses, e.g., CAN or ARINC 429, are still used to handle some specific avionics domains. Although this architecture reduces the time to market, it conjointly leads to inherent heterogeneity and new challenges to guarantee the real-time requirements.

To handle these emerging issues, a homogeneous avionic communication architecture based on the AFDX technology to interconnect different avionics domains may bring significant advantages, such as easier installation and maintenance and reduced weight and costs. Furthermore, this homogeneous communication architecture needs to support mixed-criticality applications, where safety-critical traffic (SCT), current rate constrained AFDX traffic (RC) and best effort traffic (BE) co-exist.

To achieve this aim, first, we assess the pros and cons of most relevant existing solutions vs the main avionics requirements, to support mixed-criticality applications on the AFDX network. Afterwards, the Burst Limiting Shaper (BLS) (proposed by IEEE Time Sensitive Networking (TSN) Task group) on top of a Non-Preemptive Static Priority (NP-SP) scheduler has been selected as the most promising solution.

Hence, our main contributions in this thesis are fourfold. First, we specify the extended AFDX incorporating the TSN/BLS on top of NP-SP. A preliminary performance analysis based on simulations has been conducted. These first results were encouraging to pursue this proposal.

Second, we conduct a timing analysis of the extended AFDX using Network Calculus, to compute the delay upper bounds of the different traffic classes and prove the determinism of such a solution. The preliminary performance evaluation has shown the efficiency of the extended AFDX to enhance the RC delay bounds, while guaranteeing the constraints. However, they have also highlighted some limitations of the proposed model in terms of pessimism.

Third, we introduce a second model of the extended AFDX to enhance the delay bounds tightness. Moreover, we propose a tuning method of TSN/BLS parameters to enhance as much as possible the RC timing performance, while guaranteeing the SCT constraints.

Finally, we validate our proposal through representative avionics case studies to assess its efficiency. The results show the enhancements of the RC delay bounds as well as the schedulability level of both SCT and RC traffic, in comparison to the current AFDX and Deficit Round Robin.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AFDX	Avionics Full DupleX
SCT	Safety Critical Traffic
RC	Rate Constrained
BE	Best-Effort
BLS	Burst Limiting Shaper
TSN	Time Sensitive Networking
NP-SP	Non-Preeptive Static Priority
SP	Static Priority
CPA	Compositional Performance Analysis
QoS	Quality of Service
DRR	Deficit Round Robin
VL	Virtual Link
MFS	Maximum Frame Size
BAG	Bandwidth Allocation Gap
IMA	Integrated Modular Avionics
GPA	Generalised Processor Sharing
TTE	Time Triggered Ethernet
AVB	Audio Video Bridging
AV	Audio Video
NC	Network Calculus
ILP	Integral Linear Programming
AWC	Achievable Worst-Case
WbA	Window-based Approach
CCbA	Continuous Credit-based Approach
DD	Dichotomous Deadline
HD	Heuristic Deadline

NOTATIONS

C	Link speed
MFS_i	Maximum Frame Size of flow or class i
J_f, Dl_f, BAG_f	Jitter, deadline and BAG of flow f
L_M^k, L_R^k	BLS maximum and resume credit levels of class k
L_M, L_R	BLS maximum and resume credit levels when considering a single shaped class in a single-hop network
L_M^{mux}, L_R^{mux}	BLS maximum and resume credit levels when considering a single shaped class in an output port multiplexer mux in a multi-hop network
BW^k	BLS reserved bandwidth of class k
BW	BLS reserved bandwidth when considering a single shaped class in a single-hop network
BW^{mux}	BLS reserved bandwidth when considering a single shaped class in an output port multiplexer mux in a multi-hop network
I_{idle}^k, I_{send}^k	BLS idle and sending slopes of class k, defined in Eq.(3.1) and Eq.(3.2)
I_{idle}, I_{send}	BLS idle and sending slopes when considering a single shaped class
$\Delta_i^{k,j}$	$i \in \{send, idle\}$, and $j \in \{max, min\}$ the BLS windows defined for class k in Eq.(4.7), Eq.(4.3), Eq.(4.2), and Eq.(4.5)
Δ_i^j	$i \in \{send, idle\}$, and $j \in \{max, min\}$ the BLS windows when considering a single shaped class
$p(k)$	Priority level of a class k
$p_H(k)$	BLS high priority of class k
$p_L(k)$	BLS low priority of class k
$HC(k)$	Set of flows or classes with a priority strictly higher than $p_H(k)$, i.e., $\forall j$ such as: $p_H(k) > p(j)$;

$LC(k)$	Set of flows or classes with a priority strictly lower than $p_L(k)$, i.e., $\forall j$ such as: $p_L(k) < p(j)$;
$MC(k)$	Set of flows or classes with a priority strictly between $p_L(k)$ and $p_H(k)$;
γ_k^n	Maximum service curve guaranteed for the traffic class k within node n
$\gamma_k^{bls,fluid}$	Maximum service curve of class k in the BLS node when considering fluid traffics
$\gamma_k^{bls,intui,fluid}$	Intuitive maximum service curve of class k in the BLS node when considering fluid traffics
$\beta_{k,f}^n$	Strict minimum service curve guaranteed to a flow f of class k in a node $n \in \{es, mux\}$
β_k^n	Strict minimum service curve guaranteed for the traffic class k in a node $n \in \{es, mux\}$ (end-system or output multiplexer) or component $n \in \{bls, sp\}$
$\beta_{k\in BLS,p}^{sp}$	Strict minimum service curve guaranteed to BLS class k when having the priority level p in a sp component
$\beta_k^{bls,fluid}$	Strict minimum service curve of class k in the BLS node when considering fluid traffics
$\beta_k^{bls,intui,fluid}$	Intuitive minimum service curve of class k in the BLS node when considering fluid traffics
$\alpha_{k,f}^n$	Input arrival curve of the flow f of class k in the node $n \in \{es, mux\}$ or component $n \in \{bls, sp\}$ in its path
α_k^n	Input arrival curve of the aggregated flows of class k in a node $n \in \{es, mux\}$ or component $n \in \{bls, sp\}$
$\alpha_{k,f}^{*,n}$	Output arrival curve of the flow f of class k from the node $n \in \{es, mux\}$ or component $n \in \{bls, sp\}$ in its path
$\alpha_k^{*,n}$	Output arrival curve of the aggregated flows of class k from a node $n \in \{es, mux\}$ or a component $n \in \{bls, sp\}$
UR_k	Utilisation rate of a class k at the input of a output port
UR_k^{bn}	The bottleneck network utilisation rate of a class k
n_k^{es}	Number of flows of class k generated per node es
F_k^m	set of flows of class k crossing multiplexer m
$R_{\beta,i}^j, T_{\beta,i}^j$	Guaranteed rate and initial latency of β_i^j such as $\beta_i^j = R_{\beta,i}^j \cdot (t - T_{\beta,i}^j)^+$

$Deadline_{k,f}^{end2end}$	End-to-end deadline of flow f of class k
$delay_{k,f}^{end2end}$	End-to-end delay of flow f of class k
$delay_{k,f}^{n}$	Delay of flow f of class k in a node $n \in \{es, sw, mux\}$
$delay_{k,f}^{prop}$	Propagation delay of flow f of class k
$delay_{k}^{end2end,m}$	End-to-end delay of class k with $m = BLS$ for the extended AFDX, and $m = SP$ for the current AFDX
$delay_{k}^{m,SW1}$	delay of class k in switch $SW1$ with $m = BLS$ for the extended AFDX, and $m = SP$ for the current AFDX

CHAPTER

ONE

GENERAL INTRODUCTION

"It's the job that's never started as takes longest to finish."

-J.R.R. Tolkien

1.1 Introduction

The growing number of interconnected end-systems and the expansion of exchanged data in avionics have led to an increase in complexity of the communication architecture. As a result, aircraft have gone from using point-to-point low rate data-buses such as ARINC 429 to using switched networks interconnecting dozens of systems. To cope with this trend, a first communication solution based on a high rate backbone network, i.e., the AFDX (Avionics Full Duplex Switched Ethernet) [1], has been implemented by Airbus in the A380, to interconnect critical subsystems. Moreover, some low rate data buses, e.g., CAN [2] and MIL-STD-1553-B [3], are still used to handle some specific avionics domains, such as the I/O process and the Flight Control Management. Although this architecture reduces the time to market, it conjointly leads to inherent heterogeneity and new challenges to guarantee the real-time requirements.

To deal with these emerging issues, with the maturity and reliability progress of the AFDX after a decade of successful use, a homogeneous avionic communication architecture based on such a technology to interconnect different avionics domains may bring significant advantages, such as easier installation and maintenance and reduced weight and costs.

In this first chapter, we start by presenting the context and problematic, before introducing the followed methodology to cope with this challenging issue. We finish by presenting the main contributions and the thesis outline.

1.2 Context and Problematic

Since the beginning of avionics engineering, with the fly-by-wire technology, electronic devices have conquered more and more systems in an aircraft. As a result, networks have been developed to interconnect these communication devices. Nowadays, all airliners rely heavily on data networks to fly safely, communicate with and entertain passengers, and assess the status of the aircraft itself.

The growing number of interconnected end-systems and the expansion of exchanged data in avionics have led to an increase in complexity of the communication architecture. Part of the problem was ARINC 429 data bus. The unidirectional point-to-point nature of this data bus led to increasingly complex architectures, resulting in high cost and weight.

To cope with these limitations, a first communication solution based on a high rate backbone network, i.e., the AFDX (Avionics Full Duplex Switched Ethernet) [1], has been implemented by Airbus in the A380 to interconnect essential subsystems. The resulting architecture is less complex and requires less cables, but results in data multiplexing and higher delays.

Moreover, some low rate data buses, e.g., CAN [2] or ARINC 429[4], are still used to handle specific avionics domains. Although this architecture reduces the time to market, it conjointly leads to inherent heterogeneity and new challenges to guarantee the real-time requirements.

Currently, the backbone network is a 100 Mbps AFDX network. However, a Gigabit version is under specification. Additionally, even though the ARINC 664 part 7 defines two levels of priorities, the AFDX currently on the A380 and on the A350 implements a FIFO queue and is only used for essential traffic. Thus, data with either higher or lower criticality level have their own private networks, interconnected via gateways to the AFDX[5].

This heterogeneity causes increasing delays, high complexity and costs. In fact, the AFDX was a strong step in the direction of architecture homogeneity, i.e., having a single network in an aircraft for all criticality levels. For all these reasons, we believe it is interesting to go further and define a new avionics network for mixed-criticality applications.

Additionally, we must keep in mind that our work has an industrial dimension. Developing new networks is very expensive, especially in terms of development and certification costs. The fulfilment of every safety criteria must be proved for each modification of the avionics hardware and software. Hence, each modification must be worth the resulting costs and provide a strong enhancement over the current architecture.

There are several problems to solve to define such a network. First, in the avionics context, for certification purposes, strict guarantees are defined, in terms of complexity, modularity and predictability for instance. Secondly, in the industrial context, the proposed solution must be cost-effective and the impact on current traffic, i.e. essential traffic, should be mitigated as much as possible.

Thus, this new homogeneous communication architecture, based on the AFDX technology, needs to support mixed-criticality applications, where safety-critical and best effort traffic co-exist. Hence, in addition to the current AFDX traffic profile for essential traffic, called Rate Constrained (RC) traffic, at least two extra profiles have to be handled. The first, denoted by Safety-Critical Traffic (SCT), is specified to support flows with hard real-time constraints and the highest criticality, e.g., flight control data; whereas the second is for Best-Effort (BE) flows with no delivery constraint and the lowest criticality, e.g., In-Flight Entertainment traffic.

1.3 Followed methodology

In the previous part, we have highlighted the problematic caused by the increasing traffic and heterogeneity of the network, e.g., delays, complexity, costs. Hence, our goal is to propose a new architecture for mixed-criticality applications, in order to have a single and homogeneous network. To achieve this aim, we have adopted the following methodology:

Step 1: Assessment of existing solutions vs avionics requirements

The first step is to study the avionics $needs$ due to the increase of data exchanges. This infers the avionics $requirements$ and $challenges$, which are mainly predictability, modularity, complexity and fairness. Then, to deal with these issues, we assess the pros and cons of well-known mechanisms such as Static Priority, Deficit Round Robin, and solutions such as Time Triggered Ethernet and the shapers proposed by the IEEE Time Sensitive Networking group, with reference to the avionics constraints. This will help us identify the most promising solutions compliant with the AFDX technology.

Step 2: Specification of the avionics network supporting mixed-criticality

Based on the findings of the first step, we can specify a new avionics network fulfilling the avionics requirements, while limiting the modifications of the current AFDX to reduce the costs. Then, to assess the potential of the proposed solution, a preliminary analysis based on simulations to compare its performances to the current AFDX implementing Static Priority will bring further insights on the proposed solution.

Step 3: Formal Worst-Case Timing Analysis

To cope with the predictability and certification requirements, we need to define a formal timing analysis this new network using Network Calculus, since it is framework used to certify the current AFDX. Afterwards, we need to assess the efficiency of the proposed analytical model by conducting sensitivity and tightness analyses when varying the traffic inputs and system parameters. Finally, we need to compare the performances of the proposed solution to current AFDX to assess its efficiency.

Step 4: Improving the solution performances

The idea is to start by identifying the main limitations or weaknesses of the solution and model specified in step 2 and 3. Subsequently, this leads to the definition of a new model and/or optimisation method to increase as much as possible the performances of the proposed solution, i.e., reducing the impact of higher priority on the essential traffic currently on the AFDX and increasing the traffic schedulability.

Step 5: Validation

Finally, we need to validate our proposal using representative avionics use-cases. We consider first a generic avionics case study, followed by a concrete application: adding Flight Control traffic to the AFDX. We need to test multiple scenarios to validate the potential of our proposed solution to support mixed-criticality.

This methodology followed during this thesis is illustrated in Fig.1.1.

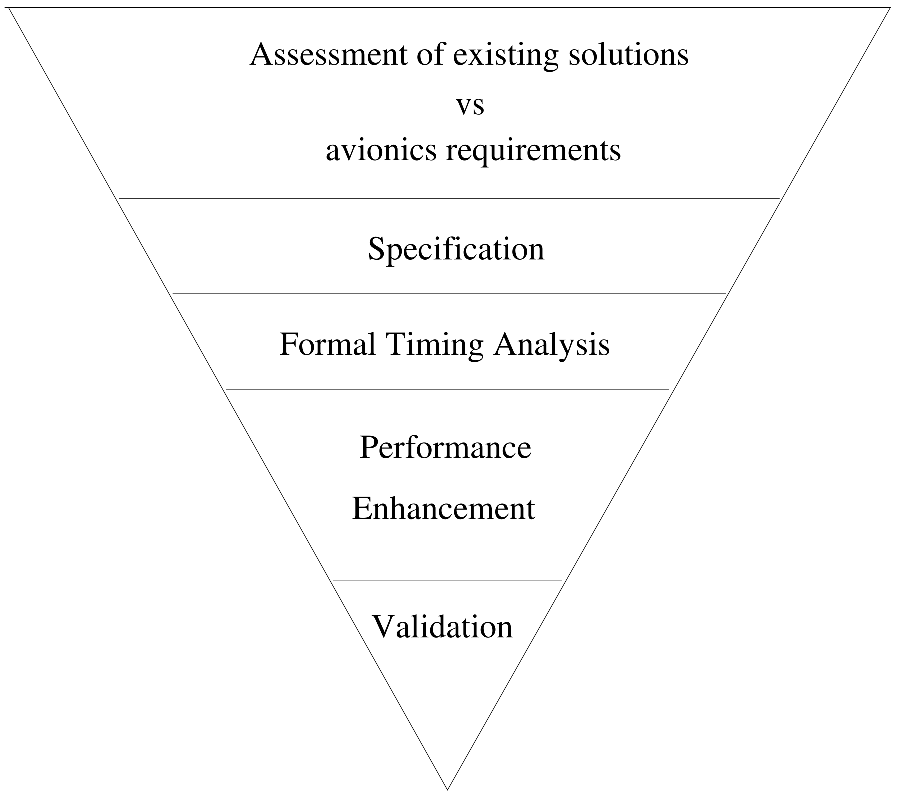


Figure 1.1: Methodology

1.4 Main contributions and outline

Our main contributions in the thesis are fourfold:

1. **The specification of an extended AFDX incorporating the TSN/BLS:** after analysing the existing solutions vs avionics requirements in Chapter 2, we have identified TSN/BLS as the most promising solution to be incorporated within the AFDX. Hence, first, we have defined the new AFDX switch architecture and an algorithm to implement the BLS. In particular, we study the necessary modifications of the switch on both hardware and software levels, including possible QoS identification mechanisms. This contribution is detailed in Chapter 3.

2. **A novel formal modelisation of this new network using Network Calculus**, in particular we have proposed a modelisation of the BLS node, based on so called sending and idle windows, denoted Window-base Approach (WbA) model. The preliminary timing analysis shows the accuracy of the model in comparison to the related work approach (CPA). Moreover, we observe noticeable enhancements of schedulability and delay bounds, in comparison to current AFDX and another solution based on Deficit Round Robin (DRR). This contribution is detailed in Chapter 4.

3. **Improving performance of the proposed solution:** first, an improved novel formal modelisation of the BLS node, based on the continuity of the BLS credit has been proposed, denoted Continuous Credit-base Approach (CCbA) model, to achieve further enhancement of the model tightness. Afterwards, we have introduced BLS parameter tuning methods to increase the solution efficiency. We have defined a first one with a low complexity but with some flaws, denoted Heuristic Deadline method. Then, we have detailed a second method, denote Dichotomous Deadline method, to fix these identified flaws at the expense of increasing the complexity. These methods enable us to conduct a fair comparison between our proposed solution and another one based on DRR. The results confirm the efficiency of our proposal. This contribution is detailed in Chapter 5.

4. **The validation of the proposed architecture:** this is detailed in Chapter 6, in particular a comparison with two other solutions: the current AFDX implementing a Static Priority (SP) Scheduler and the DRR. We use a representative avionics multi-hop architecture to validate the results of previous chapters. This validation confirms that the second model is an improvement over the first one, and corroborates the enhancements brought by the optimised BLS over both SP and DRR. In particular, we have shown that the more complex tuning method has much better results, but of course needs more computer power. Finally, we apply our proposal to a concrete avionics problem, adding the flight control traffic to the backbone AFDX. The results have highlighted the efficiency of our solution.

CHAPTER

TWO

PROBLEM STATEMENT AND STATE OF THE ART: MIXED-CRITICALITY SOLUTIONS FOR AVIONICS APPLICATIONS

"You can't stop the change, any more than you can stop the suns from setting."

-Shmi Skywalker

Contents

2.1 Introduction

Avionics is a field that moved from point-to-point transmissions to high speed networks. However, this field slowly evolves due to the stringent safety requirements and the aircraft long life expectancy, around 25 to 30 years. The comparison of this lifespan against other networking fields is an interesting one. For example, the last 30 years have seen the development of main stream Internet, from low rate 64K to high speed Gigabit fiber connections. Concerning mobile networks, a new generation appears approximatively every 9 years. Hence, between the day in 1990 when an airliner entered into service to its retirement in 2015, a consumer download link was multiplied by 15,000 and 3 mobile network generations were developed.

This highlights the stark difference between the closed avionics world, and the Internet and mobile open world. However, linkages exist between these communities: the newest avionics network, the Avionics Full-DupleX Ethernet (AFDX) is based on a technology developed for the Internet, the Switched Ethernet. The low cost and maturity, after decades of use in the consumer and industrial markets, are the main advantages of this technology.

There are still many technologies from the open world that could be used for avionics networks. The same features that attracted the AFDX designers to the Ethernet are present in other technologies. In particular in the open world, many have studied and implemented a large number of solutions allowing mixed-criticality within a network. In this chapter, we study existing solutions to assess their potential use to solve our challenge: defining a new avionics network for mixed-criticality applications.

Hence, we start by presenting the avionics context through the evolution of avionics network and the main avionics requirements. Afterwards we assess the pros and cons of the most relevant existing mixed-criticality solutions supporting mixed-criticality applications versus the main avionics requirements. Finally, we select the most promising solution guaranteeing the requirements and challenges, i.e. Predictability, Modularity, Fairness and Complexity. Hence, the identified solution is based on the Burst Limiting shaper defined by the IEEE Time Sensitive Networking task group.

2.2 Avionics Context

Since the first planes relying sorely on mechanical instruments and fly-by-cable steering, electronics have had a growing importance in piloting an aircraft. This starts with the development of electronic devices such as sensory feedback, radar, motor monitoring. Finally, even the most important function of a plane, steering, is implemented using electronics: it is the start of fly-by-wire aircraft. With the increasing need of data exchanges within the plane, new standard have been developed, starting with ARINC 419[6] in 1966. This standard and its successor, the ARINC 429 [4] (developed in 1978) are crafted, not by individual companies but collectively by almost the entire industry. This highlights the dire need for new data networks at the time. The ARINC 429 is well-established in the avionics industry[4] and it is used in most of both retired and active commercial aircraft series.

2.2.1 ARINC 429

The ARINC 429 [4] (in full MARK 33 Digital Information Transfer System), published in 1978, was one of the first standard specifically developed for aircraft. ARINC 429 uses a twisted shielded pair to connect one static sender to several static receivers (up to 19). As a consequence, a receiver cannot reply to a message through the same bus. All lines are simplex connection (even though shielded pairs are used) with a nominal throughput of 12 – 14 kbps for the low speed version, or 100 kbps for the high speed version.

Due to the simplex links and the static definition of senders and receivers, when considering *n* systems having to exchange information, as illustrated in Fig.2.1, at least *n* buses are necessary to allow each entity to reply to any other. In Fig.2.1, six Line Replaceable Units (LRUs) [7] are represented with the six buses necessary to have each LRU able to send messages to the five others.

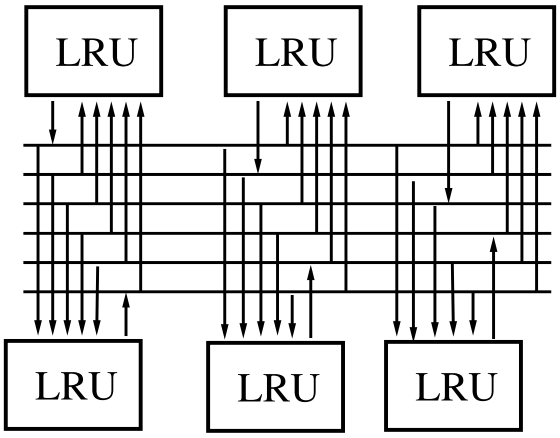


Figure 2.1: ARINC 429 network

As the number of interconnected systems grew, it became impossible to keep using such low rate point-to-point technology and new standards were developed, such as ARINC 629 and MIL-STD-1553B.

2.2.2 ARINC 629

The ARINC 629[4][8], or Digital Autonomous Terminal Access Communication, was developed by Boeing and NASA to overcome ARINC 429 limitations. This standard was mainly used in Boeing 777 with some ARINC 429 as backup. It addresses the main ARINC 429 limitations: it can connect up to 128 systems, and each system can both send and receive messages, and with a rate of 2 Mbps. As a consequence, this network is a lot more flexible than ARINC 429 and requires less cable, resulting in a much reduced complexity. However, the proposed standard uses a time-based, collision-avoidance concept in which each terminal is allocated a particular time slot to access the bus and transmit data. Each terminal determines when the appropriate time slot is available using several control timers embedded in the bus interfaces. As a consequence, collisions may occur on the bus, which decreases the determinism of the bus and causes increased latency.

2.2.3 MIL-STD-1553B

The MIL-STD-1553B bus [3][4] is a military standard published in 1973. Unidirectional connections were becoming too complex and too expensive, this is why new standards were defined, among them the MIL-STD-1553A in 1973, then the MIL-STD-1553B in 1978. This bus was first used by the US military air force inside the F-16. However, it is now broadly used and part of the conception of many military aircraft and satellites. The MIL-STD-1553 was implemented in the A350 because it is simple and less expensive to use than the 629 bus. It is worth noting that the A350 flight control network with the 1553 bus has backup ARINC 429 networks.

The high reliability[1] and capacity of 1 Megabits per second (Mbps) (which can be increased up to 200 Mbps) are the main interesting characteristics of this bus. This makes this bus very useful for military architectures. MIL-STD-1553 is a numerical half-duplex Command/response bus. This means a terminal must wait for a command from the controller before being allowed to send information on the bus. Additionally, this bus uses Time Division Multiplexing: the messages are sent on the same bus at different times. The controller manages the emission transaction table to prevent collisions. The main limitation of this technology is the limited number of interconnected systems: only 31, far from the hundreds of systems currently interconnected on a commercial aircraft. So, to connect 6 LRUs we need only one 1553 bus. However, for redundancy purposes, additional bus may be necessary to ensure strict safety guarantees as illustrated in Fig.2.2.

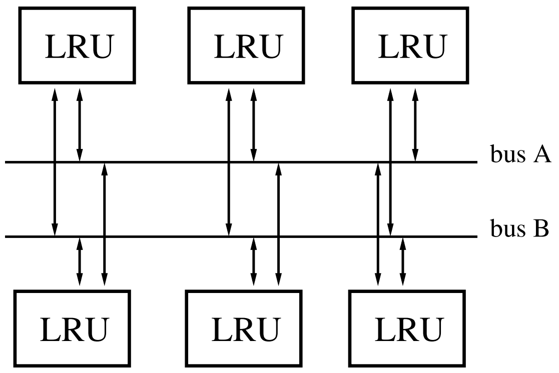


Figure 2.2: MIL-STD-1553 diagram

However, none of these avionics standards fills the emerging need of more bandwidth. The development of the A380 was the opportunity to develop a new standard customised to AIRBUS needs: the Avionics Full-DupleX Ethernet, standardised as the ARINC 664 part 7[1].

[1]Reliability of bus 1553: one erroneous word for every 10 millions correct ones, one word is 20 bit long

2.2.4 Avionics Full-DupleX Ethernet (AFDX)

To reduce the cost of such a development, the new network developed by Airbus is based on a widely used one: the Ethernet. The Ethernet has been used for decades outside the avionics industry and has proved its robustness, inexpensiveness, and flexibility. However, the non-determinism of the Ethernet is an issue. This non-determinism is due to several facts [4]: i) frames can be reorganised or manipulated due to micro-segmentation; ii) collisions may occur within the network and lead to frame loss. Thus, no strict quality of service guarantees can be enforced with Ethernet and the determinism of the new standard must be proved to be usable in an avionics setting.

Firstly, full-duplex switched Ethernet is used to increase the determinism in the AFDX[1] by preventing collisions. Secondly, segmentation is disable and static switching is implemented. With these modifications, the route of a frame and the number of hops necessary to reach a frame's destination is known. However, the network is still non-deterministic, which leads to the definition of Virtual Links and the use of shapers in the end-systems and switches to isolate and manage the different flows.

Virtual Links are virtual point-to-point connections implementing the same concept as used in ARINC 429: one sender, several receivers. Each VL is characterised by: i) a identification number $VLID$; ii) a priority iii) the Bandwidth Allocation Gap (BAG) ranging in powers of 2 from 1 to 128 milliseconds, which represents the minimal inter-arrival time between two consecutive frames; iv) the Maximal Frame Size (MFS) ranging from 64 to 1518 bytes, which represents the size of the largest frame sent during each BAG; v) the maximum jitter. The $VLID$ and the priority are used by the switch to find the correct output port and the correct queue. The MFS and the BAG define a maximum bandwidth for each VL. The maximum jitter ensures a certain flexibility concerning the arrival time of the frames, while setting a limit for the jitter.

Two *safeguards* have been defined to ensure that the characteristics of a VL are enforced and to prevent over-talkative end-systems from impacting other flows. The first is a *shaper* in the end-systems: before a frame is enqueued in an output port, the end-systems checks that the VL characteristics are fulfilled, in particular if the BAG is respected. If not, the frame is delayed. The second safeguard is a *policer* set in the switch that aims at ensuring the respect of the flow characteristics[9]. If not, then the frame is discarded. This ensures flow isolation: frames from over-talkative end-systems are discarded and so they do not delay other flows.

By fixing the VL parameters off-line and preventing changes online, the network keeps constant timing properties, which favour a deterministic behaviour.

The A380 and A350 have fully redundant AFDX networks[10]. So to connect 6 LRUs, we need to link them to both networks, as presented in Fig.2.3.

Obviously developing an entirely new network would require tremendous cost and effort. Currently only important data transit on the network, with a maximum load of 30%. Meanwhile, highly critical data and lower criticality data have their own private networks.

However, the next AFDX generation remains in an experimental state and is a Gigabit network, providing enough bandwidth to add new data to the current background data. As for th current AFDX, the next generation of AFDX will most likely be certified for highly critical data, leaving open the possibility of adding both highly critical data and low priority traffics.

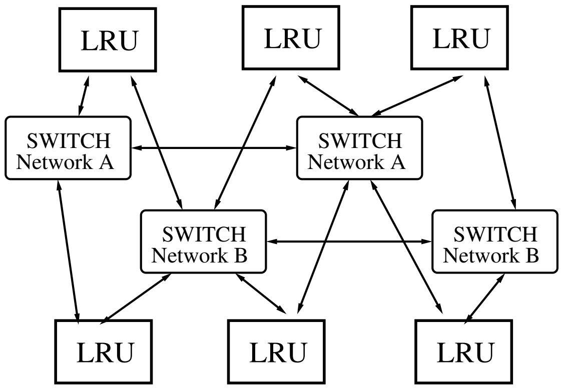


Figure 2.3: AFDX diagram

However, the impact of these new traffics on the current data should be mitigated as much as possible and the determinism should be proved.

Hence, to achieve this aim, we study the most relevant solutions compliant with a switched Ethernet network, and we mainly focus on their software aspects.

2.2.5 Avionics Requirements

In this section, we detail the different avionics requirements. From them, we identify the constraints that must be fulfilled by an avionics network. This provides a firm ground to compare the different solutions and select the most appropriate one to support mixed-criticality traffic on the AFDX.

- **Predictability** of the system: the traffic behaviours are predicable and repeatable[11]. With point-to-point network the determinism was trivial to prove, but it was one of the main challenges of ARINC 629 or the AFDX. Moreover, response must be within prescribed time period;

- **Modularity**: at first, the avionics networks had a federated architecture: it did not distinguish between hardware and software and implemented independent collections of dedicated computing resources (computing processor, communications and I/O) for each avionics function, typically contained in separate LRUs. These LRUs are thus unique to a degree and each of them needs to be fully certified.

  As there are no distinction between hardware and software, re-certification is necessary even if only the software has been updated. Additionally, components of federate networks usually cannot share resources and spare resources need to be defined for each system individually. This leads to a high degree of idle capacity, added weight and costs due to the additional resources.

  The resulting network is proprietary, has potentially extreme cable overhead, and is made of multiple different line replaceable unit fulfilling unique roles. Due to the limitations of such an architecture a new model progressively emerged: Integrated Modular Avionics (IMA). IMA discriminates between software and hardware and defines an abstraction layer between the hardware and software functionality.

As of now, IMA still uses unique LRUs but encourages the hardware homogenisation. An IMA-host can execute several software but each application must be isolated from each other. As resources are shared, spare resources can be computed collectively for all run applications.

Finally, we can define modularity as the fact that common elementary components can be configured to fit different avionic applications. This feature aims to minimise the (re) configuration and readjustment effort to facilitate system maintenance and its progress over the years;

- **Safety**: a fourth requirement is that systems must be classified according to their criticality and respect the safety rules as set by FAR Part 25.1309. Among the defined mechanism to handle this is the system partitioning. Currently, the AFDX has been certified for the highest criticality (Catastrophic) but is only used for Hazardous criticality [12].

Theses four requirements have been used to defined a standard: the ARINC 653, Avionics Application Software Standard Interface. It is the only Real-time Operating System (RTOS) that fulfils all the avionics requirements. It was developed by the air transport industry and has been adopted in 1997. Since then, all new commercial aircraft implement it. In fact, Airbus group specifically developed the AFDX to implement ARINC 653. It resulted in a partially IMA compliant network: many systems, with the same criticality, are now on the AFDX backbone. However, many other systems retained their private networks, causing additional complexity. In this thesis, we propose to go a step further by proposing a new network for mixed-criticality applications respecting the different requirements. Moreover, we identify the main following challenges to achieve:

- **Complexity** : a first challenge is the simplicity of the system. Complex systems are expensive both to implement and certify. A high level of complexity adds fault modes that have to be explored and possibly solved. Hence, as a matter of fact, solutions must be as simple as possible;

- **Fairness**: as we multiplex different types of traffics, the impact of higher priority traffic must be limited as much as possible to preserve lower priority's timing requirements.

Finally, the main avionics requirements and challenges considered here to analyse the existing solutions are:

- **Modularity**: common elementary components can be easily configured to fit different avionic applications to limit (re)configuration effort;

- **Predictability**: the impact of a system on an other is known and bounded. The communication architecture must be predictable, where the extended AFDX has to guarantee bounded latencies respecting the temporal constraints of the mixed-criticality traffic;

- **Fairness**: the impact of higher priorities on lower priorities must be reduced as much as possible while respecting timing constraints;
- **Complexity**: the complexity of implementation must be low.

Now that the avionics context has been presented, we review the different existing solutions to support mixed-criticality applications in AFDX. We compare them to the avionics requirements in order to select the most promising one.

2.3 Real Time Ethernet-compliant Solutions

Various solutions have been proposed in the literature to support mixed-criticality applications in embedded systems and particularly in avionics [13][14]. The first proposed solution was the simplest one, based on static priority and event-triggered paradigm. Overtime, new solutions with increased complexity were proposed. Some of these new solutions are based on the time-triggered paradigm. The event-triggered paradigm is known as highly flexible and facilitates the system reconfiguration, but it infers at the same time an indeterminism level and needs further proofs to verify the predictability requirement. On the other hand, the time-triggered paradigm is highly predictable, but presents some limitations in terms of system reconfigurability.

Hence, the considered paradigm is of utmost importance to quantify the reconfiguration effort needed by an alternative avionics communication architecture, in comparison to the current AFDX standard. Furthermore, the modularity level of a solution also depends on the communication paradigm.

There is a large number of solutions [15][16][17][18][19][20][21]. We reduce our presentation to the most pertinent solutions: the current solution (Static Priority), well-known fair schedulers (Deficit Round Robin and its family), and three new proposals: Time-Triggered Ethernet[22], Audio Video Bridging [18] and Time-Sensitive Networking[21].

In this section, we study the main real time Ethernet-compliant solutions and compare them to the identified avionics requirements and challenges. We also present the current state of study of each solution in reference to the main steps of our followed methodology (see Section 1.3).

2.3.1 Non-Preemptive Static Priority Scheduler

The Static Priority (SP) scheduler is the simplest QoS implementation, with very low complexity. Each queue has a defined priority and the scheduler dequeues the first frame of the eligible queue (a queue with enqueued traffic) with the highest priority, which makes the SP highly unfair. This is the scheduler defined in the AFDX standard [1], more precisely, the non-preemptive Static Priority Scheduler (NP-SP).

As NP-SP has been extensively used for decades, the scientific literacy is prolific on this subject [23][24][25][26].

In the AFDX, the predictability is enforced[1] thanks to the leaky bucket shapers in the end-systems and policers in the switches. There are many works on worst-case end-to-end delays as it is a certification requirement. The first one in [27] was done with the Network Calculus framework. Other methods, such as model checking [28][29], and Trajectory Approach [30][31][32] have been proposed to improve the delay computation and schedulability. There are even some works done on probabilistic delays [33][34].

Finally, like all event-triggered solutions, NP-SP has a good modularity. but presents some unfairness[35].

2.3.2 GPS approximations

Another type of event-triggered solution is the Generalised Processor Sharing (GPS) approximation. The Generalised Processor Sharing is an idealised scheduling algorithm that achieves perfect fairness: the capacity is shared depending on fixed weights. Many algorithms have been developed to come as close as possible to the GPS, for instance the Weighted Fair Queuing (WFQ) [16].

To approximate the fluid model of the GPS, WFQ calculates the arrival time of the first frame in each queue from the size of a frame, and the theoretical departure time calculated by the ideal GPS. WFQ then chooses the queue with the soonest arrival time and transmits the packet. The worst-case delay for WFQ, compared to GPS, is increased by the transmission time of a maximum-sized frame. However, the problem of WFQ compared to GPS is since the packet is chosen depending on its arrival time it may arrive sooner than anticipated, which creates a sizable jitter. This is why WFQ was improved to take into account the departure time through a new algorithm called Worst-case Fair Weighted Fair Queuing [36] (WF^2Q). WF^2Q results in a lessened jitter by transmitting the packet with the soonest arrival date only if the GPS departure date has been reached. This gives WF^2Q almost the same service as GPS with only an added delay of one maximum sized frame transmission time. Unfortunately, WFQ and its family are difficult to implement in hardware [37] as they require a compromise between implementation complexity and accuracy in approximating an idealised model. This drastically increases the complexity of such a hardware implementation in the AFDX switch.

Other GPS implementations in avionics are the Round Robin and its family, for example Weighted Round Robin (WRR) [38] and Deficit Round Robin (DRR) [39]. Ordinary round-robin servicing of queues can be done in constant time. With WRR, the usual implementation consists in setting a number of frames that can be consecutively sent for each queue. The major problem of this scheduler is the unfairness caused by possibly different packet sizes of the different flows [40][41]. A first solution is to adapt the weights depending on the traffic [41][42]. A counter can also be used to keep track of traffic transmitted as proposed by the Deficit Round Robin (DRR) [40]. Each round, each queue has a certain quantum of service (in bytes) assigned. When a frame is transmitted, its size is subtracted from the quantum. If the remaining quantum is not sufficient to send the next frame, it is added to the quantum received in the next round. Thus, the quantum keeps track of the deficits: queues that do not consume all their quantum in a round are compensated in the next round.

In [39], an AFDX network implementing the DRR was specified and studied. Results show the good performance of the proposal in terms of fairness. However, in [43], the trade-off between low complexity, low latency and fairness of DRR has been discussed. The results show structures such as vector trees are used to achieve a good fairness with an algorithmic complexity of $O(1)$.

Like NP-SP, GPS approximations, especially WRR and DRR have been extensively studied and used: they are implemented in most well-known simulators for instance OMNET++ or ns-2. Additionally, the schedulability and parameter tuning are well-known issues, which are still considered to be complex problems[43].

There are formal worst-case timing analyses for both WRR [44] and DRR [43][45]. In [45], a new Network Calculus model is proposed. The conclusion of their performance analysis is that the DRR is not suitable for flows with low delays and a hierarchical scheduler (with priorities) should be envisaged. In [46], an avionics architecture using DRR, is proposed but they consider video and audio flows rather than flows with tight deadlines, giving credit to the conclusions of [45].

Finally, DRR is the GPS approach which seems to be the most promising: it can be implemented in hardware without the complications due to a virtual clock, and is better for variable length traffics than WRR. Additionally, DRR is a well-known fair scheduler with several proposed modelisations.

2.3.3 Time Triggered Ethernet

TTE [22][17] is an industrial protocol developed by TTTech Computertechnik AG and is fully compliant with the Ethernet standard. This network consists of TTE switches, TTE end systems, standard Ethernet switches and standard Ethernet end-systems. TTE end systems can only be connected to TTE switches, while standard Ethernet end systems can be connected to either TTE or standard switches. The access to the medium is done through coordinated Time Division Multiple Access (TDMA). The main feature of TTE is its system-wide global time. It is fault tolerant with a good fault isolation and consistent diagnosis.

TTE defines three message formats. The first one is Time Triggered (TT) which is defined by its period, offset and length. This type is configured off-line with dedicated transmission slots and messages are dropped if they arrive outside of these slots. If a TT slot is unused, then it is freed and can be used by another traffic class. TTE has different policies for traffic integration: preemption and shuffling. In the case of non-preemptive traffic, with shuffling the transmission and relay of TT traffic may be delayed by unsynchronised frame. The second type of traffic is Rate Constrained (RC). These messages have specified rate and length, and are not sent at fixed points in time. RC frames can be queued inside the switches. Losses are avoided by calculating minimal buffer sizes offline and by limiting bandwidth with a leaky bucket algorithm. This is done in the end-system, the time between two frame emissions is measured and if a minimum gap is not reached, the frame is delayed. This class has been designed with the AFDX traffic in mind, which makes TTE a good avionics network candidate. The last class is Best-Effort (BE) which has the lowest priority, uses what is left of the bandwidth and has neither guarantee of transmission nor reception.

This solution has been entirely specified by TTTech and its design has been analysed [22]. Extensive research has been done to provide good simulation environment for TTE: a new simulation framework for TTE proposed in [47], while an extension of the OMNeT++ framework to add TTE is proposed [48]. Formal timing analysis models have also been developed, one with a Network Calculus model [49], and a model based on "busy period" [50]. Finally, the problem of schedulability and parameter tuning has also been studied, a solver reducing the NP-hard schedulability problem to a manageable size is proposed in [51].

TTE relies on a complex time table to manage the link access. So, when a flow is added, deleted or modified, this can impact the whole system through the time table. This gives TTE a very low modularity and high complexity.

Also, the aim of TTE is to guarantee the best service to the TT class and as such the full impact of the added TT traffic is felt by lower priorities (RC and BE). Thus, the fairness of TTE is also low. However, the leaky bucket and time table checks prevent over-talkative traffics from impacting other traffic, so TTE has a good predictability.

2.3.4 Audio-Video Bridging

The rapid spread of multimedia usage makes the IEEE 802.1 Audio/Video Bridging (AVB) [18] protocol of interest as it provides end-to-end delay guarantees in Ethernet networks.

This standard has been developed to provide strict timing guarantees to the media (audio and video) flows. AVB has several advantages, first, it offers a single mechanism for both audio and video flows. Secondly, the AVB offers quality control mechanisms to ensure the reliability of the transfer. This includes Quality of Service (QoS), Traffic Shaping and Bandwidth Reservation. Also on a more practical side, AVB offers an easy set up with AVB switches which automates network set up. Finally, AVB makes network convergence more easy to achieve, as AVB is an IEEE protocol compatible with current data standard.

Our objective is to multiplex different kinds of data streams, which is a kind of network convergence. To support this, we study in detail the AVB protocol to determine if it could be used in avionics case.

First, in IEEE 802.1BA published in 2009: Audio Video Bridging (AVB) Systems, a set of usage-specific traffic profiles are specified to help interoperability between networked devices using the AVB specifications.

To enforce deadlines to these AVB flows, three standards have been proposed:

- IEEE 802.1AS: Timing and Synchronization for Time-Sensitive Applications (generalized Precision Time Protocol or gPTP)
- IEEE 802.1Qat: Stream Reservation Protocol (SRP)
- IEEE 802.1Qav: Forwarding and Queuing for Time-Sensitive Streams (FQTSS)

In the next sections, we present the different features of the AVB standards.

2.3.4.1 Timing and Synchronisation

The first standard, IEEE 802.1AS, defines a Layer 2 time synchronisation service using the most stringent requirements of consumer electronics applications.

Before AVB, the used timing synchronisation had been defined in the IEEE 1588 standard, officially entitled "Standard for a Precision Clock Synchronization Protocol for Networked

Measurement and Control Systems". It is called the Precision Time Protocol (PTP). This protocol is based on a "GrandMaster clock", which is used to set a common time between the elements of the synchronised network. The selection of this grandmaster clock is based on a negotiated process that selects the network clock considered as "best", using a so-called "Best Master Clock Algorithm" (BMCA). Once it has been selected, the job of the GrandMaster clock is to send its current time to any device that requests it using "sync" packets. In order to be well-synchronised, the receiving clock must also take into account the link delay and transfer time to compensate for the transfer time of "sync packet".

However, there are a few limitations in IEEE 1558. First, even in a single switch network, the transfer time of a packet is not a constant value. The jitter is even larger in multi-hop transfers. Hence, as only the GrandMaster clock sends "sync" messages, the jitter can quickly become very large and cause clock drifts.

With IEEE 802.1AS, a "GrandMaster clock" is also used. However, it only serves to initialise the synchronisation process. While IEEE 1558 is based on an end-to-end synchronisation, from the GrandMaster clock to the different clocks, IEEE 802.1AS is based on peer-to-peer exchanges. After the first device receives its clock from the GrandMaster, it synchronises with the next AVB switch or end-device. As a result, several clocks can be updated at the same time while limiting time drift, as the clock exchanges are done with the closest neighbours.

Hence, the IEEE 802.1AS is also called generalized Precision Time Protocol (gPTP). Thanks to this new standard, the time synchronisation is much improved, with an accuracy better than $1\mu s$. This allows better synchronisation of Audio and Video flows.

Now that we have presented the time synchronisation, in the next section, we present another important AVB feature: bandwidth reservation.

2.3.4.2 Stream Reservation Protocol

Stream Reservation Protocol is a key AVB feature, described in IEEE 802.1Qat. A talker on a AVB network advertises its stream and specifies the necessary bandwidth to the switch it is connected to. Then, all the switches in the network become aware of the stream availability and the required bandwidth.

If a listener wants to receive a stream on the network, it can request to receive the stream to the nearest switch. The switch then determines whether bandwidth is sufficiently available for the new stream without overloading the output port. The request is passed on to all the switches in the stream path. When this stream is determined to be achievable (i.e. all the output ports in the path have sufficient bandwidth), the bandwidth required is reserved across the entire path. It is worth noting that the amount of bandwidth that can be reserved for Audio Video (AV) stream is limited to a certain quantity adjustable within AV switches.

With this bandwidth reservation, the network participates in the media exchange and the reservation prevents the overloading of the network, protecting the existing media flows and automating part of the management of the network. Finally, by limiting the maximum reserved bandwidth, AVB ensures that a certain amount of bandwidth is left for data transfer, participating in the network convergence previously mentioned.

2.3.4.3 Forwarding and Queuing: the Credit Based Shaper

In IEEE 802.1Qav, the Forwarding and Queuing for AV bridges provides a mechanism to split time-critical and non-time-critical traffics into different classes. The time-critical traffic can then be applied to the credit-based shaper (CBS).

AVB specifies the Credit-based Shaper (CBS) algorithm for real-time (RT) traffic classes "A" and "B". Each traffic class uses dedicated queues, so scheduling within a class follows a FIFO order. The aim is to guarantee a fixed maximum latency for up to 7 hops within the network for class A and B: 2 ms for A and 50 ms for B. This is done by preventing traffic bursts thanks to a credit-dependant shaping.

Each shaped class has a credit-counter managing a gate allowing or forbidding the frame transmissions. The credit is consumed (decreased) at the constant rate (the send slope) when data of the specific class is transferred, or else replenished (increased) at a different constant rate (the so-called idle slope) when a shaped frame is enqueued. Depending on the credit value, the gate associated to the queue is open (when the credit is positive or null) or closed (when the credit strictly negative). When the queue is empty, the credit immediately returns to 0.

The idle slope and send slope are configured using the Stream Reservation Protocol (SRP) as defined by IEEE 802.1Qat: the reserved bandwidth depends on the bandwidth needed by the considered stream.

After the CBS, the different classes are scheduled using a static priority scheduler (see Fig.2.4). Hence, CBS prevents the starvation of lower priorities, while ensuring bandwidth guarantees to the shaped queue. These are good properties for mixed-criticality applications.

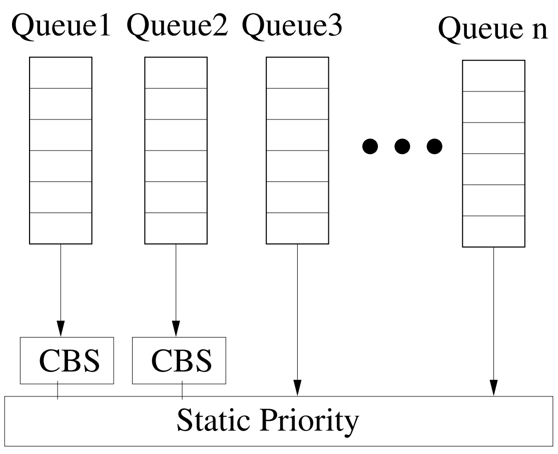


Figure 2.4: Credit Based Shaper

2.3.4.4 AVB related research

Extensive research work has been done on the AVB as it has gathered the interest of the automotive, automation and aeronautics communities. Among this work, many simulation results have been presented, such as [18][52], showing the high potential of the AVB for automotive networks. In [53] and [54], the ability of the AVB to deal with the real-time constraints typically found in the industrial automation was assessed. Finally, concerning the avionics applications, in [55], the reliability of AVB was evaluated and results showed that it could only be used for low safety classes, which demand less reliability. This is mainly due to the complexity inherent to automation of bandwidth reservation and the failure probability of devices evaluated in [55]. In [56], the AVB is compared to the ARINC 664 (AFDX). The results again highlight the need for further work to enable the AVB as a standard for avionics. However, the authors note the possibility of implementing the CBS shaper on the AFDX.

Quite a number of worst-case timing analyses have been done, with different methods. There are examples using delay computation [53], Network Calculus framework [57], Trajectory Approach [58], and Modular Performance Analysis (MPA) [59]. These four methods make easy the computation of a class worst-case delay.

With the formal worst-case timing evaluation in [60], the response time of each individual message is proposed. This work has generally shown that Ethernet AVB substantially increases the latencies for the highest-priority Class-A traffic compared to the static-priority arbitration, due to the additional traffic shaping delay. We also observe in these evaluations that class B traffic can not benefit from the shaping of class A as it is shaped itself. However, work presented in [61] shows that the analysis done in [60] only considers one blocking effect, from lower priority messages. But with AVB, this is not always the case due to traffic shaping. So a new response time analysis is presented in [61]. The proposed analysis is limited to the constrained deadline traffic model and a single-switch architecture. In [62], the [61] model is extended to include multi-hop networks, where the effect of over-reservation is also studied.

Concerning the avionics requirements, a consequence of the shaping of classes A and B with CBS is that the worst-case latency of unshaped lower priorities is improved. The shaping allows giving minimum bandwidth guarantees to lower priority traffic. Hence, this shaper validates the fairness condition.

Concerning the predictability, the different classes are isolated from each other thanks to the counters and their associated blocking effects when the gate is closed, i.e., the credit is strictly negative. The use of VL from the AFDX specification can also isolate the non-shaped traffics from each other. However, it is shown in [61] that the impact of the blocking effect of the AVB on the latency is high, giving a medium predictability to this shaper: the respect of the deadlines is uncertain due to the high latency caused by the CBS.

Among all these works, one recurring conclusion is the high latency caused by the AVB shaper. The main drawback of the AVB/CBS is the fact that it is by nature a blocking shaper: frames cannot be transmitted if the credit is strictly below 0, no matter the state of the other queues. This can cause unnecessary delays if other queues are empty.

To try and solve this issue, several approaches such as fragmentation and preemption, are studied in [63]. However, in our avionics setting, such solutions are deemed too complex to implement. Other solutions using the second generation of the AVB, called Time Sensitive Network (TSN) have been explored and will be explained in the next section.

2.3.5 Time-Sensitive Networking

As presented in the previous section, the main limitation of the AVB is the high latency due to the shaping. Hence, in the AVB generation 2, a new traffic class with very strict timing guarantees has been proposed.

To limit the impact of the lower priorities on higher priority traffic, two standards have been published: IEEE Std 802.1Qbu-2016 Frame Preemption and IEEE Std 802.1Qbv-2015 Enhancements for Scheduled Traffic. The first one allows a Bridge Port to suspend the transmission of non time-critical frames during the transmission of critical frames. The second one specifies time-aware queue draining to schedule the transmission of frames relative to a known time scale.

An other standard has been published to provide explicit trees for data traffic: IEEE Std 802.1Qca-2015 Path Control and Reservation.

Finally, there are several on-going draft by the TSN task group.

- 802.1AS-Rev - Timing and Synchronisation: Timing and Synchronisation for Time-Sensitive Applications - Revision
- 802.1CB - Frame Replication and Elimination for Reliability
- 802.1Qcc - Stream Reservation Protocol (SRP) Enhancements and Performance Improvements
- 802.1Qch - Cyclic Queuing and Forwarding
- 802.1Qci - Per-Stream Filtering and Policing
- 802.1Qcj - Automatic Attachment to Provider Backbone Bridging (PBB) services
- 802.1CM - Time-Sensitive Networking for Fronthaul
- 802.1Qcp - YANG Data Model
- 802.1Qcr - Asynchronous Traffic Shaping

In the following section, we detail the different TSN standards. Then in the next section, we present the different shapers proposed in TSN. Among all the standard, two do not present an interest to our study: 802.1CM - Time-Sensitive Networking for Fronthaul and 802.1Qcp - YANG Data Model. The detail of these two standards are informational only and can be skipped. The other parts of TSN offer interesting possibilities in terms of delay management and network configuration such as preemption, queue management [64], automatic bandwidth reservation.

2.3.5.1 TSN standards

In this section we present the different TSN standards, published and drafted. SO if a TSN solution is selected, we will be able to take into account the full scope of the possibilities offered.

IEEE Std 802.1Qbu-2016 Frame Preemption

The purpose of this standard is to reduce the latency transmission for scheduled, time-critical frames in a bridged LAN.

A large, non time-critical frame may start before time-critical frame transmission. This condition leads to excessive latency for the time-critical frame. The lack of transmission preemption severely limits the capabilities of implementing a real-time network with applications using scheduled frame transmission.

IEEE Std 802.1Qbu also provides the discovery, configuration, and control of preemption service for a bridge port and end station, while ensuring that preemption is only enabled on a given link if both link partners have that capability.

This standard defines a class of service for time-critical frames that requests the transmitter in a bridged Local Area Network to suspend the transmission of a non-time-critical frame, and allow for one or more time-critical frames to be transmitted. When the time-critical frames have been transmitted, the transmission of the preempted frame is resumed. A non time-critical frame could be preempted multiple times.

To achieve this, the preemption is done at the MAC layer, after two processes of frame selection, one for the non-preemptable queue (called express), and one for the preemptable queues. Both process go through MAC control before being merged in a MAC merge sublayer. Meanwhile, the physical layer remains ignorant of the preemption. Finally, guard band can protect the express traffic completely from interferences from preemptable traffic: the time reserved for scheduled traffic is increased to take into account guard bands, which prevents the preemption time from impacting the scheduled frames.

IEEE Std 802.1Qbv-2015 Enhancements for Scheduled Traffic

Bridges are increasingly used to interconnect devices that support scheduled applications (e.g., industrial automation, process control and vehicle control). This standard provides performance assurances of latency and delivery variation to enable these applications while maintaining the existing guarantees for the credit-based shaper and best-effort traffic.

This standard specifies time-aware queue-draining procedures, managed objects and extensions to existing protocols that enable bridges and end stations to schedule the transmission of frames based on timing derived from IEEE Std 802.1AS: the transmission from each queue is scheduled relative to a known timescale using the global time. A transmission gate is associated with each queue and the state of the gate determines whether or not queued frames can be selected for transmission. The detailed of the Time Aware Shaper defined in this standard will be presented in Section 2.3.5.2.

802.1AS-Rev - Timing and Synchronisation: Timing and Synchronisation for Time-Sensitive Applications - Revision

This standard enables stations attached to bridged Local Area Networks (LANs) to meet the respective jitter, and time synchronisation requirements for time-sensitive applications. To facilitate the widespread use of bridged LANs for these applications, synchronisation information is one of the components needed at each network element where time-sensitive application data are sent or a time-sensitive function is performed. This standard uses the work of the IEEE 1588 Working Group by developing the additional specifications needed to address these requirements.

802.1AS-Rev specifies the protocol and procedures used to ensure that the synchronisation requirements are met for time-sensitive applications, such as audio, video, and time-sensitive control, across the whole network. This includes the maintenance of synchronised time during normal operation and following addition, removal, or failure of network components and network reconfiguration. It also specifies the possibility of using IEEE Std 1588 specifications.

The revision includes an improved scalability through a one-step processing, and the improved support for long-chains and rings. The new standard is also more responsive with faster Grand Master changes, and a reduced Best Master Clock Algorithm convergence time. The redundancy of GrandMaster and paths are also possible.

802.1CB - Frame Replication and Elimination for Reliability

This standard specifies procedures, managed objects and protocols for bridges and end stations that provide the identification and replication of frames, for redundant transmission, the identification of duplicate frames and the elimination of duplicate frames.

802.1Qcc - Stream Reservation Protocol (SRP) Enhancements and Performance Improvements

This amendment provides protocols, procedures and managed objects for bridges and end stations that are compatible with existing automatic reservation AVB mechanisms and provides:

- Support for more streams
- Configurable SR (stream reservation) classes and streams
- Better description of stream characteristics
- Support for Layer 3 streaming
- Deterministic stream reservation convergence
- UNI (User Network Interface) for routing and reservations

802.1Qch - Cyclic Queuing and Forwarding

This amendment specifies synchronised cyclic enqueuing and queue draining procedures, managed objects, and extensions to existing protocols that enable bridges and end stations to synchronise their transmission of frames to achieve zero congestion loss and deterministic latency.

802.1Qci - Per-Stream Filtering and Policing

The development of standards for Time-Sensitive Networking (TSN) have shown that there exist no interoperable standards that enable a bridge to detect whether or not some systems in a network are conforming to behaviours agreed by configuration and/or protocol exchanges. For example, devices that exceed the allocated bandwidth for one stream can prevent the network from achieving the benefits of TSN for any or all streams, not just the misbehaving stream.

This standard specifies procedures and managed objects for a bridge to perform frame counting, filtering, policing, and service class selection for a frame based on the particular data stream to which the frame belongs, and a synchronised cyclic time schedule. Policing and filtering functions include the detection and mitigation of disruptive transmissions by other systems in a network, improving the robustness of that network.

802.1Qcj - Automatic Attachment to Provider Backbone Bridging (PBB) services

This standard specifies the protocols, procedures and management objects for auto attachment of network devices to Provider Backbone service instances by using Type, Length, Value (TLVs) within the Link Layer Discovery Protocol (LLDP).

This also simplifies the deployment and administration of PBB networks, e.g. controlled by Shortest Path Bridging (SPB), by allowing for automatic configuration of the virtual LANs and service identifiers, thus allowing access to services of network devices without the need of manual configuration.

802.1CM - Time-Sensitive Networking for Fronthaul

Fronthaul is a new mobile architecture. The fronthaul portion of a Centralized Radio Access Network (C-RAN) telecommunications architecture comprises the intermediate links between the centralised radio controllers and the radio heads (or masts) at the "edge" of a cellular network.

A mobile operator's radio equipment and radio equipment controller are often separated and the connection between them has very stringent requirements. This fronthaul connection is not provided by a bridged network today. In an IEEE 802.1 bridged network potentially carrying other categories of traffic, specific configurations of various IEEE 802 standards are needed to meet the requirements of the fronthaul streams. Therefore, the use and the configurations of functions defined in the IEEE 802 standards have to be specified by standard profiles for bridged fronthaul networks.

This standard defines profiles that select features, options, configurations, defaults, protocols and procedures of bridges, stations and LANs that are necessary to build networks that are capable of transporting fronthaul streams, which are time sensitive.

802.1Qcp - YANG Data Model

802.1Qcp specifies a Unified Modeling Language (UML) based information model and a YANG data model that allows configuration and status reporting for bridges and bridge components including Media Access Control (MAC) Bridges, Two-Port MAC Relays (TPMRs), Customer Virtual Local Area Network (VLAN) Bridges, and Provider Bridges. It further defines the relationship between the information and data model and models for the other management capabilities specified in this standard.

YANG (Request For Comment (RFC) 6020) is a formalised data modeling language that can be used by NETCONF, a widely accepted protocol that is being used to simplify network configuration. Other standards development organisations (e.g. Internet Engineering Task Force (IETF) and the Metro Ethernet Forum) have adopted YANG, and are developing a broad range of data models.

802.1Qcr - Asynchronous Traffic Shaping

There is well defined traffic that requires zero congestion loss and deterministic latency. Current bridging standards do not provide a sufficiently fine grained asynchronous traffic mechanism to meet these requirements without using network topology information. The draft specifies mechanisms that do not rely on synchronous communication, thereby providing independence from clock synchronisation mechanisms and higher link utilisation than synchronous mechanisms. The proposed mechanism is called Urgency-based scheduler (UBS) and will be detailed in section 2.3.5.2.

Hence, the TSN task group aims at improving the AVB standards with 802.1AS-Rev and 802.1Qcc, but they also seek to add new features such a preemption and frame replication. Now that we have presented the different parts of TSN, we present in more details the shaping solutions proposed by the TSN task group. TSN first proposed 3 shapers: Time Aware Shaper (TAS), Peristaltic Shaper (PS) and Burst Limiting Shaper (BLS). Among these 3, TAS was selected for the standard 802.1Qbv and uses the time synchronisation defined in 802.1AS and 802.1AS-Rev. After this, a last mechanism was proposed for the 802.1Qcr standard, the Urgency-based scheduler (UBS). In the next 4 sections we present them all.

2.3.5.2 TSN shapers

In this section, we present the four shapers proposed by the TSN task group, starting with the Time Aware Shaper.

Time Aware Shaper

TAS[65] uses time-driven scheduling to manage link access between traffic classes which makes it a good candidate for mixed-criticality traffic. The global time used by TAS is defined in 802.1AS and 802.1AS-Rev. As illustrated in Fig.2.5, for each traffic class, the scheduler at an output port contains a time-aware gate per queue, which allows frames to pass when opened and blocks frames when closed. The times T_n at which these gates open (o) and close (C) are programmed offline (gate schedule). Gates of multiple traffic classes can be opened at the same time. Then, dequeuing is arbitrated according to the priority of these classes.

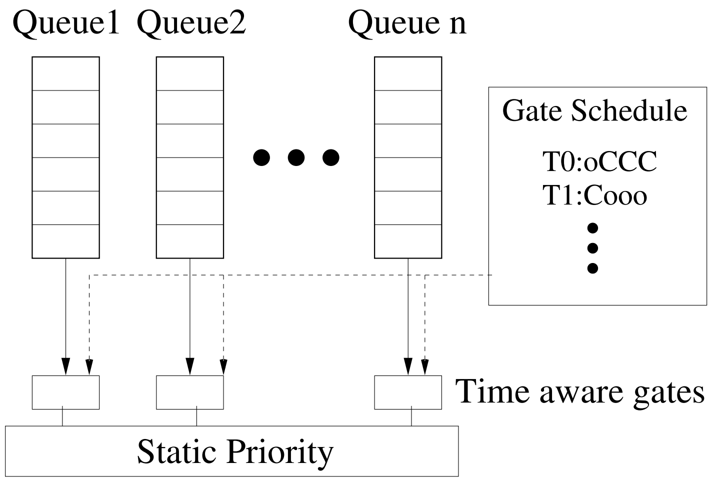


Figure 2.5: Time Aware Shaper

To prevent frames transmission after its gate is closed, TAS defines guard bands. From the start of a guard band until the gate is closed, no new frames of the corresponding class are allowed to start transmission. The idea behind TAS is that each critical traffic class has privileged link access, i.e., every critical traffic class has exclusive link access during its intervals, i.e., without interference by higher or lower-priority traffic. However, same-priority traffic can still impact a frame.

TAS has been specified by the TSN Task Goup[21] and even before the specification, in [66], simulations have been run to compute end-to-end latencies in an automotive setting. Concerning the formal analysis, a formal modelisation using the Compositional Performance Analysis (CPA) methodology is proposed in [65]. In [67], the schedulability is studied, the authors explore the constraints linked to the gate opening and closing, which is a first step toward finding a formal method of schedulability.

TAS is very close to TTE in terms of goals and how to achieve them. Due to the gate schedule, modifications are propagated to all flows, giving TAS a low modularity and high complexity.

As TTE, the aim of TAS is to reduce the impact of lower classes. Additionally, when lower classes's gates are opened, they are scheduled using a Static Priority. A solution could be to use time-synchronisation on several classes to reserve for them a certain bandwidth. However, this comes with other problems, such as bandwidth over-reservation due to use of guard-bands to limit the impact of non-preemption. With guard-bands, part of the bandwidth cannot be used, resulting a higher impact on the asynchrone traffics. Thus, TAS does not have a good fairness.

Finally, while TAS does not define bandwidth guards for asynchrone traffics, these already exist on the AFDX. So with the use of VL, TAS has a good predictability.

Since the publication of the standard, numerous works have been published concerning this new shaper and its impact on AVB traffics [62], [68], in the automotive industry [69][70], and also in the mobile fronthaul network [71][72][73].

Peristaltic Shaper

The Peristaltic Shaper (PS) [74] uses a global time divided in odd and even phases to manage different traffic classes. If a shaped frame arrives in an odd (resp. even) phase, it can not be sent before the start of the next even (resp. odd) phase. The idle time can be used by other priorities.

The Peristaltic Shaper has been proposed by the same task group as TAS. Hence, before the standardisation, they have often been studied together and similar work has been done such as simulations and formal models [66] [65].

As for TTE and TAS, the use of a global time has a strong impact on the modularity: a flow modification can impact the calculation of odd and even phases not only on its path, but on others too as the phase modification impacts other flows. So, PS has a low modularity and high complexity.

Due to the initial waiting time caused by the odd and even phases, it is possible that lower priority traffic is sent faster than with a Static Priority scheduler. Hence, Peristaltic Shaper is slightly fairer than Static Priority, making it an interesting solution.

Concerning the predictability, with the VL, the Peristaltic Shaper has a good predictability.

Now that the time-triggered solutions have been presented, we detail the event-triggered solutions.

Burst Limiting Shaper

The Burst Limiting Shaper (BLS) belongs to the credit-based shaper class. Presented in [15], the BLS is always based on a static priority scheduler, as it modifies the priority seen by the SP depending on a credit counter.

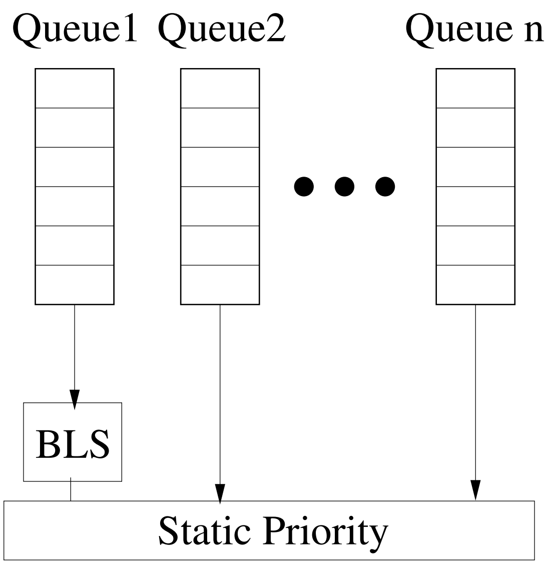


Figure 2.6: Burst Limiting Shaper

Hence, depending on the priority value, the shaped frames can be blocked or not by other classes. However, no matter the state of the credit, the first frame with the highest priority is sent. Thus, contrary to CBS, the BLS is a non-blocking shaper, which is a large improvement for the predictability of the solution. The BLS also has a low complexity thanks to the simple counter and priority change.

The priority change is the feature that enables the BLS to reserve bandwidth for the shaped queue. It also has an other consequence: the limitation of the bandwidth available to the shaped queue, which make the association of BLS and SP much fairer than the SP alone.

The Burst Limiting Shaper has been studied. In particular simulation results and timing analysis are available in [66] [75].

Urgency-based Scheduler

The main idea of the Urgency-based Scheduler (UBS) [76] is a separation between per flow and per queue. The conceptual separation of per flow queue and state provides per flow shaping at every hop for flow aggregated in the queues. This concept is called $interleaved\ shaping$. This significantly reduces the algorithmic complexity by limiting the number of required queues. Hence the first step when a new frame arrives in the output port is to select the appropriate queue depending on the priority of the flow and its "urgency" as decided by an interleaving algorithm.

This results in a two level queuing mode: first an array of shaped queues outside of the scheduler, each associated to a fixed priority and an individual shaper and secondly one queue per priority level in the scheduler. Each of the latter queues merges the output of the all shaped queue of the same priority level. The queues outside the scheduler are called $queues$ and the queues in the scheduler, $pseudo\ queues$. The term $pseudo\ queues$ is based on an implementation assumption: if only a few (shaped) queues are necessary, efficient scheduler implementation are possible that directly transmit from the (shaped) queues. The next packet for transmission is identified by comparison of (i) the associated priority levels, (ii) the eligibility times of the Head-of-Queue (HQ) packets (which depends on an interleaving algorithm [76]). This has motivated the introduction of new theory of traffic regulators able to explain the “reshaping-for free” property of minimal interleaved regulators [77]. In [78], the UBS synthesis problem of assigning hard real-time data flows to queues and priorities to queues is presented and solved. However, while very easy to implement on a software level, this seems quite complex to implement at the hardware level or at the MAC level. Additionally, the selection of an appropriate queue for each packet again increases the complexity and adds potential points of failure.

This is summed up in Fig.2.7: when a new frame arrives in the output port, it is dispatched in a queue depending on a queue allocation scheme. Then, each (shaped) queue is shaped by an interleaving algorithm before merging same priority queues into a single queue using the time stamp, computed by the interleaving algorithm. Finally, a static priority scheduler selects the appropriate packet. From this, the complexity of the solution and the many points of failure are quite obvious.

This scheduler is still new, so little research has been done yet. In [79], a first analysis of the integration of UBS in a fault tolerant system is done. The aim is to find an optimum architecture with minimal cost. In [76], the scheduler is presented, simulations and timing analysis are performed. The results show high link utilisation and low delays. They also conclude that the implementation complexity is low, in part because they assume the queue selection process is already implemented in the switches thanks to the standardisation of 802.1Qci-Per-Stream Filtering and Policing. But as showed here, while implementing it in higher layer is simple, implementing at the hardware level is much more complex.

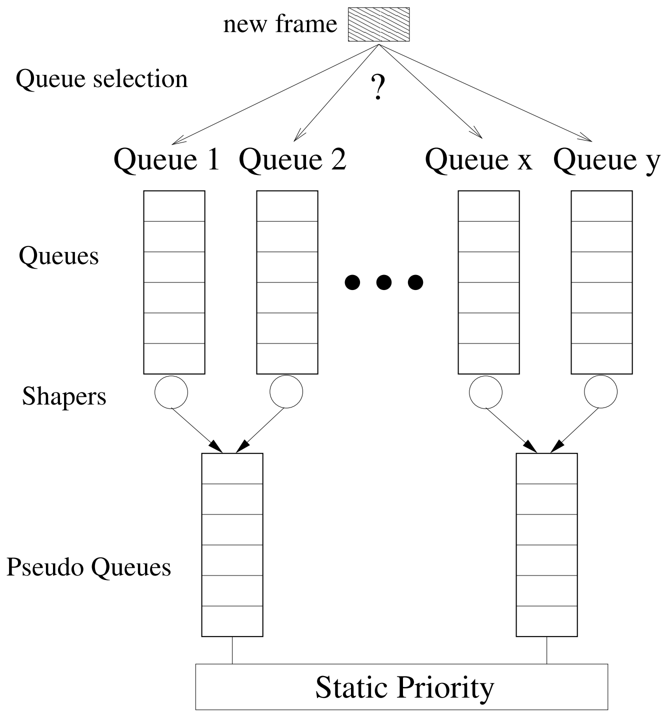


Figure 2.7: Urgency-based Scheduler

Now that we have presented and studied the different solutions to add mixed-criticality to the AFDX, in the next section we compare these solutions and select the most promising one regarding the avionics requirements and challenges.

2.4 Discussion

From the presentations of the different mixed-criticality solutions, we can select the solutions fit for an avionic context.

In this section, we assess the pros and cons of the different solutions vs the four avionics requirements and challenges, to find the eligible ones:

- **predictability:** thanks to the leaky bucket shapers in the AFDX end-systems and the policers in the switches, all the presented solutions can achieve the necessary determinism and isolation;

- **modularity:** the event-triggered paradigm is known for its modularity, contrary to the time-triggered event paradigm. Because of the time-synchronisation, a modification of a flow can impact other flows directly and necessitate a recomputation of the gate schedules. Hence, the time triggered solutions do not fulfil the modularity criterion, contrary to event-triggered solutions;
- **fairness:** there are four solutions fulfilling the fairness constraint: DRR, CBS, PS and BLS.
- **Complexity:** time-triggered solutions necessitate implementing a complex time synchronisation. Both CBS and BLS can be used independently from the synchronisation aspect of AVB and TSN. Concerning UBS, while is also asynchronous, we showed that its complexity is nevertheless high.

Finally, the only eligible fair solutions are the DRR, the AVB/CBS and the TSN/BLS.

However, AVB/CBS has two main issues: i) even when the other queues are empty, if the credit is negative a shaped frame is blocked; ii) the credit returns to 0 when the queue is empty, erasing any potential positive impact due to the idleness of previous frames. This causes unnecessary delays, so we also discard CBS. Interestingly, the BLS solves both theses limitations as it is a non-blocking shaper with a continuous credit.

The DRR is a well-known scheduler that has been used and studied extensively by many communities, among them the avionics community [39]. But, it infers high complexity due to its parameters, i.e., weights, tuning process. BLS however, is a new shaper, mainly studied by the automotive community [75]. Nonetheless, it has also started gaining attention from the avionics community [80].

One of the interesting feature of the BLS is its ability to shape one queue and leave the others to SP. A simple DRR however, shapes all the queues, reserving bandwidth for lower priority traffic; whereas the BLS lets the non-real time traffic use the remaining bandwidth left by real-time traffic.

Therefore, the BLS is considered herein as the most interesting solution to be incorporated within the AFDX standard, to enable an homogeneous avionics communication architecture supporting mixed-criticality applications.

As the BLS was proposed with TSN, we now study the possibility of using other TSN features. First, we discard the time synchronisation as too complex and not useful since we selected an event-triggered solution. The different standards requiring time synchronisation, such as 802.1Qch-Cycle Queuing and Forwarding and 802.1Qbv Enhancement for Scheduled Traffic, must be discarded too. The standards adding automation (802.1Qcj and 802.Qcc) to the network cannot be used as they add unpredictability to the network. Finally, the remaining three features are frame preemption (802.1Qbu), frame replication (802.1CB), and per-stream filtering and policing (802.1Qci). The last two are already implemented in the AFDX so we discard them. Concerning frame preemption, it is a very interesting feature but it is complex to implement, so we discard it. Thus, the BLS is the only feature we selected to specify the new avionics network.

The conclusions on the considered solutions vs the main avionics requirements and challenges are illustrated in Table 2.1.

Solutions	references	Constraints				Key studies			
NP-SP	[25] [81]	✓✓	✓✓	X	✓✓	✓	✓	✓	✓
GPS/DRR	[43] [45] [44] [39]	✓✓	✓✓	✓✓	✓	✓	✓	✓	✓
TTE	[22] [47] [17] [50] [51]	✓✓	X	X	X	✓	✓	✓	✓
AVB/CBS	[60] [82] [61]	✓	✓✓	✓✓	✓✓	✓	✓	✓	✓
TSN/TAS	[65] [66]	✓✓	X	X	X		✓	✓	
TSN/PS	[74] [66] [65]	✓✓	X	✓	X		✓	✓	
TSN/BLS	[66] [15] [75]	✓✓	✓✓	✓✓	✓✓		✓	✓	
TSN/UBS	[76] [79] [78]	✓✓	✓✓	✓✓	X		✓	✓	

Avionics constraints and challenges	Predictability
	Modularity
	Fairness
	Complexity
Specification	
Timing Analysis	Simulation-based
	Analytical-based
Performance enhancement	

Table 2.1: Existing solutions vs avionics requirements and challenges

2.5 Conclusion

After analysing the existing solutions, it appears clearly that time-triggered solutions are not fit for avionics due to their high complexity and low modularity . Among the event-triggered solutions, DRR, CBS and BLS are all possible solutions, but the BLS seems the best one, as it is less complex and has better predictability.

Now that we have selected the most promising solution, we will follow the methodology described in Section 1.3. In particular, since the specification of the BLS has not been completed yet, we will start by specifying our proposed solution in Chapter 3. It will contain the specification of an AFDX switch incorporating the BLS and an algorithm of such an implementation.

CHAPTER
THREE

SPECIFICATION: THE EXTENDED AFDX SWITCH

"Without requirements or design, programming is the art of adding bugs to an empty text file."

-Louis Srygley

Contents

3.1 Introduction

The development of avionics data exchanges led to study the possibility and the impact of mixing different criticality levels within a single network. Introducing mixed-criticality in a network is a hot topic that can be handled using different solutions ranging from well-known scheduling schemes (e.g., SP, WRR,...) to newly proposed shapers (e.g., CBR, BLS,...). Among this myriad of solutions, we have shown in Chapter 2 the potential of the Burst Limiting Shaper (BLS) proposed by the Time Sensitive Networking (TSN) task group, part of the 802.1 working group, in the specific case of an avionics network.

Hence, the aim of incorporating the TSN/BLS within the AFDX switch is to handle mixed criticality data, and more specifically three AFDX traffic profiles, as illustrated in Fig.3.1: (i) the Safety-Critical Traffic (SCT) with the highest criticality and the tightest temporal deadline, e.g., Flight-control flows; (ii) Rate-Constrained traffic (RC) with the medium criticality and a deadline constraint to guarantee, e.g., the current AFDX flows; (iii) the best-effort traffic (BE) with the lowest criticality and no time constraint, e.g., in-Flight Entertainment.

The aim of this chapter is to detail the specification of such a solution. We start by detailing the BLS basic concepts. This is a necessary step to first gain a better understanding of the BLS before proposing an implementation. Secondly, the extended AFDX switch architecture is presented to highlight the necessary modifications of the current one. Finally, a preliminary analysis is done to emphasise the interest of the BLS to enhance performances.

As a result, this chapter presents the first contribution of this thesis, which is the specification of an extended AFDX implementing the TSN/BLS on top of Non-preemptive Static Priority (NP-SP) scheduler, along with a preliminary analysis. The study shows the BLS is straightforward to implement and that the current AFDX switch architecture implementing a NP-SP scheduler requires few modifications to implement the BLS. Additionally, our preliminary simulations show the potential of such a solution with improvements of both maximum utilisation rates and delay bounds of RC traffic.

3.2 The Burst Limiting Shaper

In this section, we first present the functioning of the BLS, then we propose an algorithm to implement it in Section 3.2.2.

3.2.1 Basic concepts

The BLS belongs to the credit-based shapers class. Each shaped queue is associated to a class k and has been defined in [15] by an upper threshold L_M^k, a lower threshold L_R^k, such as $0 \leqslant L_R^k < L_M^k$, and a reserved bandwidth BW^k. Additionally, the priority of a class k shaped by BLS, denoted $p(k)$, can vary between a high and a low value , denoted $p_H(k)$ and $p_L(k)$ (with priority 0 the highest priority, and $p_L(k) > p_H(k)$). The low value is usually below the lowest priority of the unshaped traffic. In the avionic context, to guarantee the safety isolation level between the different traffic profiles, the low value associated to the SCT is set to be lower than the RC priority level, but higher than the BE priority. Therefore, when considering one class for each traffic type, SCT queue priority oscillates between 0 (the highest) and 2 (see Fig.3.1), RC priority is 1 (see Fig.3.1) and BE has the priority 3 (the lowest, see Fig.3.1). Thus, when SCT traffic is enqueued, BE traffic can never be sent no matter the state of BLS. In this case, RC is the only traffic that can be sent and this only happens when the SCT priority is 2. As a consequence, BE traffic is isolated from SCT and RC traffics.

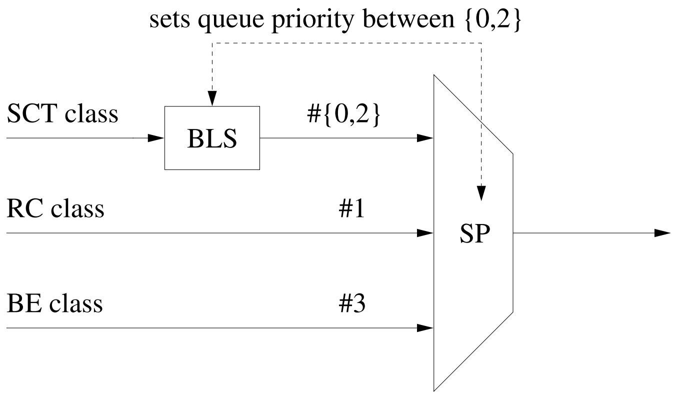


Figure 3.1: Burst Limiting Shaper on top of NP-SP at the output port with 3 classes

The credit counter varies as follows:

- initially, the credit counter starts at 0 and the priority of the queue of the burst limited flows is high (#0);
- the main feature of the BLS is the change of priority $p(k)$ of the queue of the shaped class, which occurs in two contexts: 1) if $p(k)$ is high and credit reaches L_M^k; 2) if $p(k)$ is low and credit reaches L_R^k;
- when a frame is transmitted, the credit increases (is consumed) with a rate of I_{send}^k, else the credit decreases (is gained) with a rate of I_{idle}^k;
- when the credit reaches L_M^k, it stays at this level until the end of the transmission of the current frame;
- when the credit reaches 0, it stays at this level until the end of the transmission of the current frame (if any). The credit remains at 0 until a new BLS frame is transmitted.

The behaviour of the BLS is illustrated in Fig.3.2. As shown, the credit is always between 0 and L_M^k. The credit rates of the BLS shaper are defined as follows:

- the decreasing rate is:

$$I_{idle}^k = BW^k \cdot C \tag{3.1}$$

  where C is the link speed and BW^k is the percentage of bandwidth reserved for BLS frames.

- the increasing rate is:

$$I_{send}^k = C - I_{idle}^k \tag{3.2}$$

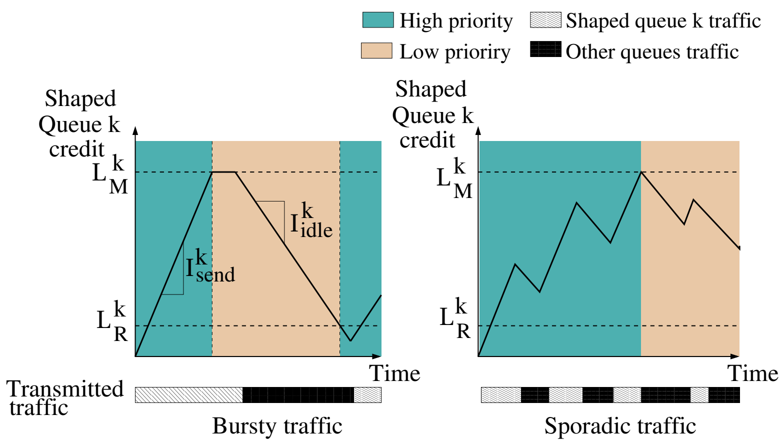


Figure 3.2: BLS credit evolution

It is worth noting that with the BLS, both the priority of the queue of the shaped class and the state of all the queues, i.e., empty or not, define whether the credit is gained or lost. This aspect is depicted in Fig.3.2 for two arrival scenarios. The first one (left figure) shows the case of a bursty traffic, where the maximum of traffic shaped by the BLS is sent when its priority is the highest. Consequently, the other priorities send as much traffic as possible when the priority of the BLS class has the low value. The second one (right figure) is for sporadic traffic, where we can see that when the shaped-class priority is highest but no frame is available, then the credit is regained. However, when the priority is at the low value and the other queues are empty, then shaped-class frames can be transmitted and the credit is consumed.

3.2.2 Implementation

The implementation of BLS at the hardware level requires a counter to track the credit and a timer to handle credit updates. These parameters, i.e., a counter and a timer, induce low extra complexity to implement a BLS on top of a NP-SP scheduler, in comparison to a regular NP-SP scheduler. Hence, the algorithm allowing to implement the BLS corresponds to a modification of the priority scheduler. The new dequeuing algorithm is presented in Algorithm 1. This algorithm operates in two cases: 1) if a frame arrives when all queues are empty; 2) at the end of the current frame transmission, a new frame has to be elected for dequeuing.

The credits of each class k is stored in $credits[k]$. Each shaped class k has a dequeuing timer. Likewise, for each class, L^k_M, L^k_R, BW^k, $p_L(k)$ and $p_H(k)$ are stored in LMs, LRs, BW^ks, pLs and pHs. A class k shaped by BLS is characterised by the fact that $pLs[k] > pHs[k]$ (as priority 0 is the highest priority), otherwise $pLs[k] = pHs[k]$. The current priority of a class is stored in p. We suppose that several class can be shaped and no two classes can have the same priority. All the timestamps used in the algorithm are set to the time value at the start of execution. Also, $timerDQs[k]$ represents the estimated end of the shaped frame transmission.

The credits $credit$ and the dequeuing timers $timerDQs[k]$ are initialised to zero. The initial priority is set to the high value. First, we store the current time in $time$ in line 2. Then, for each BLS class k (line 1), we compute δ_{time}, the difference between the current time and the time stored in $timerDQs[k]$ in line 3. The duration δ_{time} represents the time elapsed since the last credit update, during which no shaped packet was sent, we call this the idle time. Then, if $\delta_{time} > 0$, the credit is updated by removing the credit gained during the idle time that just occurred (lines 4 and 5). Next, $timerDQs[k]$ is set to the current time to keep track of the last time when the credit was updated (line 6). If the credit reaches L_R^k, the priority changes to its high value (lines 7 and 8). Then, with the updated priorities, the priority scheduler performs as usual: each queue is checked for dequeuing (lines 2 and 13). When a BLS queue is selected, the credit expected to be consumed is added to the $credit$ variable (line 16). The time taken for the packet to be dequeued is added to the variable $timerDQs[k]$ (lines 16 and 17). Therefore, the transmission time of the packet will not be taken into account in the idle time δ_{time} (line 3). If the credit reaches L_M^k, then the priority changes to its low value (lines 18 and 19). Finally, the packet is dequeued (line 22), and the loop is exited in line 23.

Algorithm 1 BLS algorithm: dequeuing process

```
Input: $credits$; $timerDQs$;$C$ $LMs$;$LRs$;$BWs$;$pLs$; $pHs$;
1: for each class $k$ with $pLs[k] > pHs[k]$ do
2:   $time = getcurrentTime()$
3:   $\delta_{time} = time - timerDQs[k]$
4:   if $\delta_{time} > 0$ then
5:     $credits[k] = \max(credits[k] - \delta_{time} \cdot BWs[k] \cdot C, 0)$
6:     $timerDQs[k] = time$
7:     if $credits[k] \leqslant LRs[k]$ and $p[k] = pLs[k]$ then
8:       $p[k] = pHs[k]$
9:     end if
10:  end if
11: end for
12: for each priority level $pl$, highest first do
13:   if length(queue(pl))>0 then
14:     k=queue(pl)
15:     if $pLs[k] > pHs[k]$ then
16:       $credits[k]$=min($LMs[k]$,$credits[k]$+size(head(k) $\cdot$(1 $- BWs[k]$)))
17:       $timerDQs[k]$=$time$+size(head(k))/C
18:       if $credits[k] \geqslant LMs[k]$ and $p[k] = pHs[k]$ then
19:         $p[k] = pLs[k]$
20:       end if
21:     end if
22:     dequeue(head(k))
23:     break
24:   end if
25: end for
```

Algorithm 1 also implements the following functions:

- $getcurrentTime()$ uses a timer to return the current time;
- $queue(pl)$ returns the queue associated to the priority pl;
- $head(k)$ returns the first packet in the queue of class k;
- $size(f)$ returns the size of the packet f;
- $dequeue(f)$ activates the dequeuing event of packet f.

The complexity of this algorithm is the same as a priority scheduler and is $O(1)$, since the number of queues is constant.

3.3 Switch architecture

In this section, we present the switch architecture. First we explain the overall view of the switch, then we detail the different solutions for QoS identification to select the best one.

3.3.1 Overall architecture

Now that BLS and its implementation has been presented, we explain here the detail of the proposed extended AFDX switch architecture.

The AFDX standard manages the exchanged data through the Virtual Link (VL) concept. This concept provides a way to reserve a guaranteed bandwidth for each traffic flow as explained in Chapter 2. Furthermore, the AFDX supports a NP-SP scheduler based on two priority levels within switches to enable the QoS features.

For the new extended AFDX, we consider that depending on the constraints of the flows, the different traffics can be separated in several classes: $\{SCT_1,...SCT_n\}$,$\{RC_1,...RC_m\}$, and $\{BE_1,...BE_l\}$, with n, m, l, the number of classes for each type of traffic. Additionally, also depending on the different constraints, any class can be shaped by a BLS at the output port as shown in Fig.3.4 to reduce the impact of the considered class on lower priorities.

In Fig.3.3, we illustrate the architecture of our extended AFDX switch in the case of 3 classes. It consists of: (i) store and forward input ports to verify each frame correctness before sending it to the corresponding output port; (ii) a static configuration table to forward the received frames to the correct output port(s) based on their VL identifier; (iii) the output ports can handle $k = m + n + l$ priority queues, multiplexed with a NP-SP scheduler, as illustrated in Fig.3.4.

As a consequence, for each queue, we associate a class k, and we can set two different priorities: $p_L(k)$ and $p_H(k)$. The BLS is only activated if $p_L(k) > p_H(k)$ (because the priority increases when p(k) decreases, i.e., priority 0 is the highest priority). In this case, a credit counter monitoring the SP dequeuing process is attributed to this queue. The credit manages the selection of the priority of the queue viewed by the NP-SP as described in Fig.3.5.

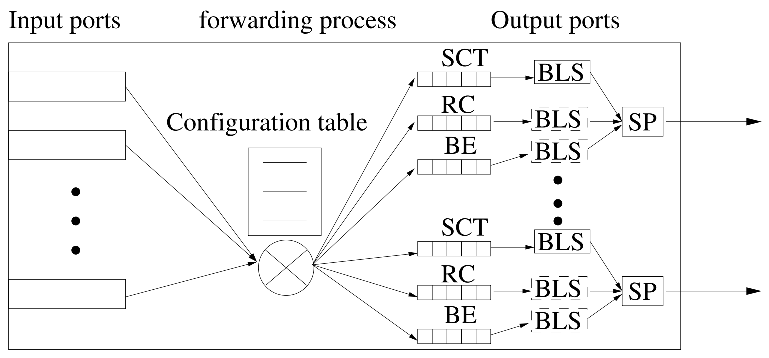


Figure 3.3: An extended AFDX switch architecture with 3 classes

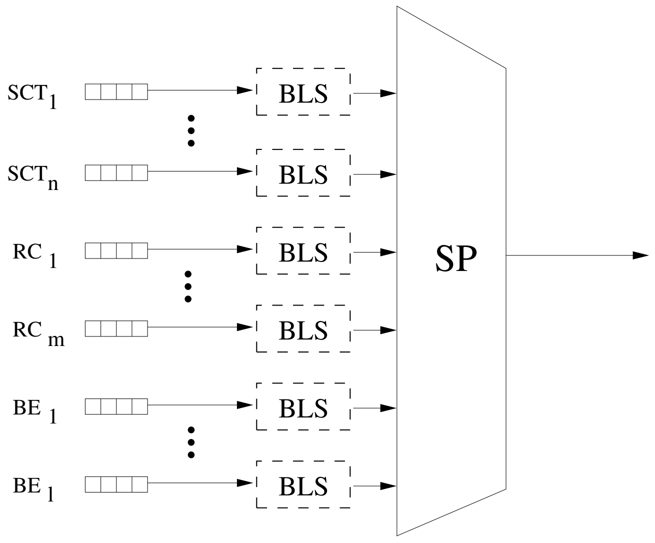


Figure 3.4: The output port of an extended AFDX switch

The resulting architecture is very flexible and offers many opportunities to manage each class as needed. For example for homogeneous classes, we can only consider one queue by type of traffic and we can shape the SCT as proposed in Fig.3.1. Or, for more heterogeneous classes, we can use two queues by type of traffic and only shape the second queue of both SCT and RC-type traffic classes, leaving the first ones for tighter deadlines, as illustrated in Fig.3.6

The current AFDX switch distinguishes the flow priority level based on its VL identifier stored in the static configuration table, i.e., for each VL identifier, there is a predefined priority level stored in the table. However there are other solutions available. We will study them in the next section. Hence, to manage both extra AFDX profiles, i.e., SCT and BE, within our extended AFDX switch, we need to update the configuration table to add the corresponding VL identifiers and their associated priority levels. Moreover, we need to update the QoS identification to implement at least 3 priorities.

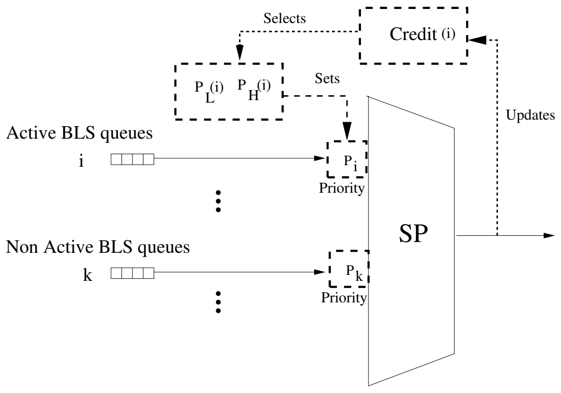


Figure 3.5: BLS behaviour in an output port of an extended AFDX switch

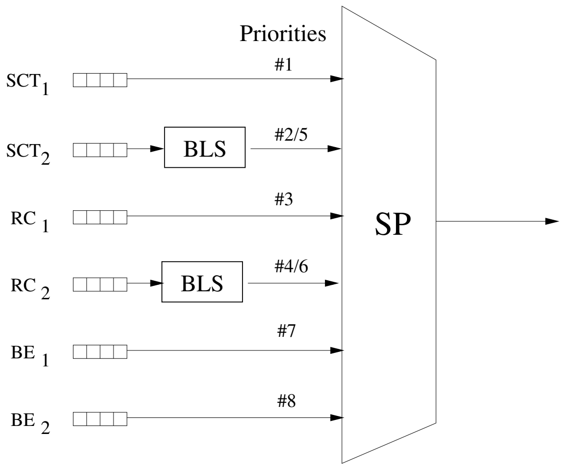


Figure 3.6: Example of an output port of an extended AFDX switch

3.3.2 QoS identification

In order to implement Quality of Service (QoS), the first problem is the identification of the class of a frame. The AFDX already uses a system to differentiate two classes of service. Currently two priorities are implemented, low and high, with 2 queues in each output port. Only one is used, leaving one free. It is enough for adding the Flight Control traffic but leaves no space for adding other traffics. In this part, we explore the different possibilities to add more priorities.

3.3.2.1 Potential solutions

Configuration Files
In the AFDX, the characteristics of a VL are defined in a configuration file shared by every switch in the network, called Filtering _Policing _and _Forwarding_ Configuration_Table. Its last column, denoted “prioritisation”, defines the priority (“high” or “low”) of a VL. We propose to modify this field to add other possibilities by adding new priority qualifiers, or using numbers to define the priority. Since "prioritization" is the last item of a line, it will not displace other fields in the line even if its length is increased, i.e., it will only change the line size. The drawback of this is the possible change of type of the “prioritisation” information, and possibly a slight growth of the configuration table file. The advantage is that no modification is necessary to the current AFDX frames. However, some modifications may be required within the switch to interpret differently the configuration file.

MAC Address
A second solution consists in using part of the constant field of the MAC address to encode the priority. This would slightly decrease the size of the configuration table since the prioritisation field could then be deleted. However, it requires changing the End-Systems and switches to build the MAC address field and guarantee its correct interpretation in the configuration table.

802.1Q
Another solution consists in using the 802.1Q header [83]. In the Tag control information (TCI) field, the Priority code point (PCP) is a 3 bits field used to define the priority of a frame, which offers 8 possibilities. Unfortunately, while this solution is appealing due to the use of a well known and globally used standard, it has the same drawbacks as the MAC address solution, i.e., required changes within End-systems and switches and no real advantage compared to the current implementation.

IP Header
A fourth solution is using the Differentiated Services Code Point (DSCP) [84][85], a field used in the IP header to differentiate the different classes. This solution is based on layer 3 of the OSI model, while the current switches only use layer 2 fields. It would mean accessing a higher thus more complicated OSI layer. Similarly to the two previous solutions, the current

AFDX frames would have to be modified in order to incorporate the assigned priority. Moreover, since the third layer is more complex, it might also be more difficult and more costly to obtain the certification of the switches.

3.3.2.2 Discussion

The various alternatives are compared in Table 3.1 according to 3 criteria:

- complexity: it takes into account the modifications needed for the switch, the End-Systems, the frame structure and the frame layer accessed by the switch;
- scalability: it is measured using the number of available classes;
- performance: it depends on the induced overhead.

The solution using the current configuration file is the one that does not require the modification of the switch, the End-Systems, or the frame, thus it has the lowest complexity. Moreover, it has a good scalability in terms of number of classes since any number could be added to the file. It also has good performances in terms of overhead, with only one column needed in the configuration table. With the other solutions, the way a switch identifies the frame class is very different. They do not use the configuration file at all and store the class inside the frame. Hence, these solutions are more complex because they require more modifications than the one based on the configuration file. Both the MAC and IP addresses use already existing field, unlike the 802.1Q which needs more modifications of the AFDX frame and consequently more overhead. However, the MAC address and the 802.1Q are less complex than the IP Address since they are layer-2 fields. Finally, 802.1Q is the less scalable solution because the number of classes is limited to 8, whereas the others can have several thousands of classes due to their field lengths.

	Config. file	MAC address	802.1Q	IP address
Complexity	++	+	+	–
Scalability	+++	+++	+	++
Performance	++	+++	+	+++

Table 3.1: QoS identification solution comparison

Hence, extending the current way to set priorities in the AFDX network seems the simplest solution since it necessitates only few modifications, and does not need access to a higher OSI layer. Moreover, with this configuration file, new and old AFDX switches could be in the same network, with a different Filtering_Policing _and _Forwarding _Configuration _Table for each type, to take into account the number of queues in each output port.

In comparison to the current AFDX switch architecture, the main modifications required for the proposed extended AFDX switch consists in:

- at the **software level**, updating the static configuration table to manage at least three priority levels instead of two (note that the update overhead is very limited since only one additional bit per line is necessary to have 4 priorities);
- at the **hardware level**, adding the necessary extra priority queues at the output port since the current AFDX switch only supports two priorities; and implementing the BLS on top of the NP-SP scheduler, as illustrated in Fig.3.4.

From the global avionics communication architecture point of view, our extended AFDX necessitates the update of the End-Systems at the application layer to enable a consistent mapping between VL identifiers and the appropriate priority level. Moreover, the implementation and certification of this extended AFDX may imply extra costs, in comparison with the current one. However, this fact is counterbalanced by the major pros of such an homogeneous architecture, in terms of enhancing performance and reducing cables and weight.

Now that we have presented the extended AFDX, we use the proposed algorithm to implement the solution on a NS2 simulator to assess the potential of our solution through simulations.

3.4 Preliminary Analysis

We present in this section the results of a preliminary analysis, which aims to show through simulations that the extended AFDX with BLS solution has promising results. While we have specified an architecture with any number of priorities, this analysis is done considering 3 classes, one for each type of traffic. First, we present our case study. Then we discuss the simulation results.

3.4.1 Case study

We consider a Gigabit switch described in Fig.3.7, and with the input traffic described in Table 3.2. The switch is connected to 4 Gigabit cables for each type of input traffic, and one Gigabit-cable for the output traffic. The number of flows of a class k enqueued in an output port, denoted n_k^{in}, determines the load of the output port. We denote UR_k the utilisation rate of class $k \in \{SCT, RC, BE\}$, which directly depends on n_k^{in}:

$$UR_k = n_k^{in} \cdot \frac{MFS_k}{BAG_k}$$

For this preliminary analysis, we consider 2 scenarios described in Table 3.3. The aim of scenario 1 (resp. 2) is to get a first idea of the impact of increasing the SCT (resp. RC) utilisation rate on RC and SCT delay bounds. In particular, we want to verify the real-time requirement, i.e., the deadlines are fulfilled when varying the load of the network; in addition to the fairness challenge, i.e., the impact of SCT on the RC in terms of delay bounds is limited.

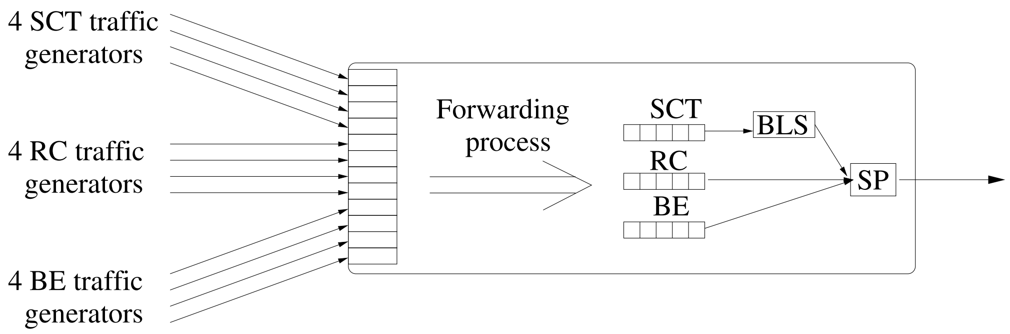


Figure 3.7: Considered extended AFDX network

Thus, in scenario 1 (resp. 2), we set RC (resp. SCT) input rates at 20%, which means generating 156 (resp. 790 flows). Then, we vary SCT (resp. RC) utilisation rate, denoted UR_{SCT} (resp. UR_{RC}) from 0 to over 70%. BE is used to bring the load up to 100% and we do not present its timing results as BE does not have a deadline.

As there is only one shaped class, SCT, we use $k = \emptyset$ to simplify the notations for the BLS parameters L_M^k, L_R^k, BW^k. Throughout the chapters, we do this when considering this use-case, for the non-ambiguous notations, such as I_{send}^k or $\Delta_{idle}^{k,min}$ for instance.

The BLS parameters are the same in both scenarios:

- $L_R = 0$ bit;
- $L_M = 22,118$ bits;
- $BW = 0.46$.

Hence, L_R is set to its minimum value, L_M is set to absorb a burst of 80 frames and BW is just below its median (0.5) value.

A scenario is defined by the input traffics and by the BLS parameters. So, we define a vector to describe our scenarios:

$$Scenario = [UR_{SCT}(\text{in }\%), UR_{RC}(\text{in }\%), L_M(\text{in bits}), L_R(\text{in bits}), BW]$$

So, for scenario 1 and 2 this gives:

$$Scenario_{SCT} = (UR_{SCT} \in [0.1:0.1:78], UR_{RC} = 20, L_M = 22,118, L_R = 0, BW = 0.46)$$

$$Scenario_{RC} = (UR_{SCT} = 20, UR_{RC} \in [0.5:0.5:72], L_M = 22,118, L_R = 0, BW = 0.46)$$

We have simulated our extended AFDX switch incorporating the BLS on top of the NP-SP scheduler and the current AFDX switch, which implements the regular NP-SP scheduler. For our case study, the simulation supports the three priority levels, based on NS2 tool. Each conducted simulation has a duration of 5s, which represents up to 3.2 millions SCT and RC simulated frames. The results of scenarios 1 and 2 are presented in Figures 3.8 and 3.9, respectively.

Priority	Traffic type	MFS (Bytes)	BAG (ms)	deadline (ms)	jitter (ms)
0/2	SCT	64	2	2	0
1	RC	320	2	2	0
3	BE	1024	8	none	0.5

Table 3.2: Avionics flow Characteristics

Scenarios	Scenario 1	Scenario 2
$(UR_{SCT};UR_{RC})$(%)	([0.1..78];20)	(20;[0.5..72])
$(n_{SCT}^{in};n_{RC}^{in})$	$([4:160:3044];156)$	$(780;[4:40:564])$
$(BW;L_M;L_R)$	(0.46;22,118;0)	(0.46;22,118;0)

Table 3.3: Parameters considered for testing scenarios 1 and 2

3.4.2 Simulation results

Impact of varying SCT utilisation rate

The delay bounds of SCT and RC when varying the SCT utilisation rate are presented in Fig.3.8. We can see that the SCT delay bound is increased by the BLS (see Fig.3.8(a)), comparatively to the regular NP-SP scheduler. In fact, after an initial sharp increase, the increase of the SCT delay bound has the same increase rate with our extended AFDX proposition and current AFDX. This is due to the BLS parameters chosen: our extended AFDX is made of two parts, a BLS and a SP, and depending on the BLS parameters and the traffic flows, one is predominant on the other. This is confirmed by the RC delay bounds (see Fig.3.8(b)): below 16%, the current and extended AFDX curves are overlapping: the SP part is predominant. After 16% they separate, showing that BLS has now a stronger impact. While the delay bound with current AFDX soars, it remains constant with our extended AFDX. This shows the good isolation provided to RC by the BLS. In fact, while the BLS increases the SCT delay bound by 0.7ms, it reduces the RC delay bound by 4ms. As a result, the RC delay bound is much reduced with our extended AFDX, while the SCT delay bound is only slightly increased. It is also worth noting that with current AFDX the RC deadline is reached at 54% while it is never reached with our extended AFDX. Thus, the maximum utilisation rate is improved by 48% (from 54% to 80%) under the proposed scheduler.

Impact of varying RC utilisation rate

The delay bounds of SCT and RC when varying the RC utilisation rate are presented in Fig.3.9. As before, we can see that the SCT delay bound is increased by the BLS (see Fig.3.9(a)), while the RC delay bound is either improved or identical. While the increase is sizable (128%), the SCT delay bound remains well below its deadline. Additionally, we can see that with the chosen BLS parameters, the BLS has a stronger impact for low values of RC: in Fig.3.9(b), there is a gap between the RC delay bound with current and extended AFDX. This gap decreases as RC utilisation rate increases. The reason is when the RC rate increases, the impact of the BLS on RC traffic decreases until it becomes negligible and only SP rules the RC delay bound behaviour. This shows the RC delay bound can be improved by the BLS, even when the BLS parameters are intuitively set. At the current utilisation rate of the AFDX (30% on the 100 Mbps AFDX network, so 3% on a Gigabit AFDX) the gain in terms of delay bound with our extended AFDX compared to the current AFDX for RC traffic is around 40%. This gain is still over 17% for an utilisation rate RC at 15% of the capacity.

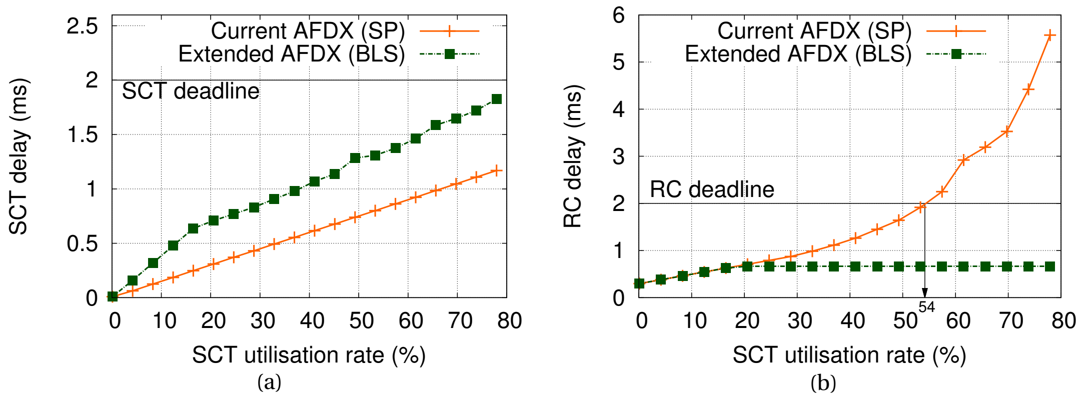


Figure 3.8: Scenario 1: impact of SCT max. utilisation rate on: (a) SCT delay bounds; (b) RC delay bounds, with $(UR_{SCT} \in [0.1:78], UR_{RC} = 20, L_M = 22, 118, L_R = 0, BW = 0.46)$

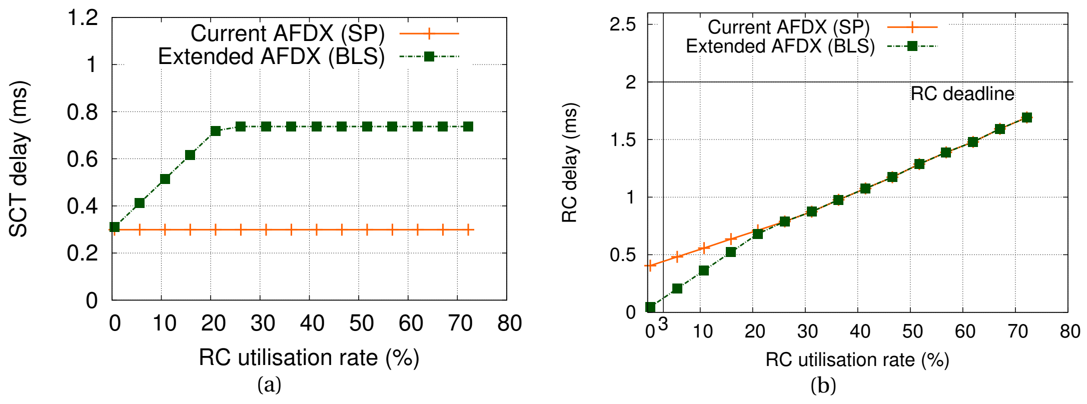


Figure 3.9: Scenario 2: impact of RC max. utilisation rate on: (a) SCT delay bounds; (b) RC delay bounds, with $(UR_{SCT} = 20, UR_{RC} \in [0.5:72], L_M = 22, 118, L_R = 0, BW = 0.46)$

These results show the ability of our extended AFDX switch to favour the predictability of the mixed-criticality traffic, which is one of the key avionics requirements. Moreover, our extended AFDX switch offers good fairness property since it enables a noticeable RC latencies decrease while guaranteeing the SCT deadline.

Now that the potential of this solution has been assessed, the next chapter presents a formal timing analysis to prove its predictability, a key requirement for avionics.

3.5 Conclusion

In this chapter, we have specified our extended AFDX, incorporating the BLS shaper on top of NP-SP, considered as the most promising solution to support mixed-criticality applications. This specification has detailed the BLS implementation and highlighted the few necessary modifications at the software as well as the hardware levels, to extend the current AFDX switches to incorporate the BLS. Finally, we have conducted simulations to evaluate the ability of our proposal to guarantee the predictability requirement, while favouring the fairness property. Results show the noticeable enhancement of the latencies of the current AFDX traffic (RC) in presence of the highest priority one (SCT) under our extended AFDX, with reference to the current AFDX.

As a next step, we will conduct formal worst-case timing analyses to compute the worst-case latencies and prove the predictability of such a promising solution to fulfil the certification needs.

CHAPTER

FOUR

FORMAL WORST-CASE TIMING ANALYSIS

" A theory has only the alternative of being right or wrong. A model has a third possibility: it may be right, but irrelevant."

-Manfred Eigen

Contents

4.1 Introduction

As presented in Chapter 2, the avionics networks have to be certified through the proof of determinism, i.e., the worst-case delay bounds fulfil the timing requirements. This fact necessitates a formal analysis of the network timing performances and particularly the BLS impact.

In this chapter, after presenting the existing works in this area and their limitations, we conduct a worst-case timing analysis of the extended AFDX (specified in Chapter 3) using Network Calculus. We finish with a preliminary performance evaluation of the proposed solution, considering the case study in Section 3.4.1.

Results show that the proposed model copes with the limitation of the state of the art approaches (CPA approach) while being tighter and more scalable. Finally, a sensitivity analysis highlights the importance of wisely choosing the BLS parameters, in particular the reserved Bandwidth BW.

4.2 Related Work: Worst-case Timing Analysis of TSN/BLS Shaper

In this section, we present the existing work on the BLS formal analysis. Then, we detail the limitations of the main one, the Compositional Performance Analysis (CPA) model.

There are some interesting approaches in the literature concerning the worst-case timing analysis of TSN network, and more particularly BLS shaper. The first and seminal one in [86] introduces a first service curve model to deduce worst-case delay computation. However, this presentation published by the TSN task group has never been extended in a formal paper. The second one has detailed a more formal worst-case timing analysis in [66], which also has some limitations. Basically, the proposed model does not take into account the impact of either the same priority flows or the higher ones, which will clearly induce optimistic worst-case delays. The last and more recent one in [75] has proposed a formal analysis of TSN/BLS shaper, based on a CPA method. This approach has handled the main limitations of the model presented in [66]; and interesting results for an automotive case study have been detailed. However, this method necessitates extensive computation power to solve two maximisation problems, an Integral Linear Programming (ILP) problem and a fixed point problem. Additionally, in some situations that will be detailed in Section 4.2.1, the CPA model can lead to optimistic delay bounds because the worst-case computation is based on the classic hypothesis that all classes are backlogged during the worst-case scenario. We will show in Section 4.2.2 that this fact may provide optimistic delay bounds; thus false guarantees for messages that will actually miss their deadline in the worst-case.

4.2.1 CPA model

The CPA model [75] computes the impact of the other flows by dividing them in four categories: the lower-priority blocking, the same-priority blocking, the higher priority blocking, and the BLS shaper blocking. The latter is defined as follows for a flow of class I:

$$I_i^{SB}(\delta t) = \lceil \frac{\delta t}{t_I^{S-}} \rceil \cdot t_I^{R+}$$

with:

- L_R^I, L_M^I and I_{idle}^I BLS parameters of class I;
- $t_I^{R+} = \lceil \frac{L_M^I - L_R^I}{I_{idle}^I} \rceil + \max_{j \in lp(I)} \frac{MFS_j}{C}$, with MFS_j the Maximum Frame Size of flow j, $lp(I)$ the streams with a priority lower than I: the maximum blocking time, called the replenishment interval
- $t_I^{S-} = \max\left\{ \left\lfloor \frac{L_M^I - L_R^I}{I_{send}^I} \right\rfloor, \max_{j \in I} \frac{MFS_j}{C} \right\}$ the shortest service interval for class I.

We have identified three main limitations in the CPA model, which may lead to over-pessimistic delay bounds, or worse, optimistic delay bounds, as it will be showed in Section 4.2.2. The first limitation concerns the maximum replenishment interval t_I^{R+}. The additional frame transmission $\max_{j \in lp(I)} \frac{MFS_j}{C}$ consider all the priorities lower than I. This computation considers two implicit assumptions, which are not necessarily fulfilled in the general case. The first implicit hypothesis is to consider that the priority for I is the BLS high priority. The second implicit hypothesis is the fact that the low BLS priority is the lowest one. The delay caused by $\max_{j \in lp(I)} \frac{MFS_j}{C}$ is due to the transmission of a frame while the BLS priority is low, just before the credit reaches the resume level. But only classes with a priority higher than the low BLS priority can be transmitted while BLS frames are enqueued, thanks to the Static Priority Scheduler. Thus, CPA model considers that all the flows are in $lp(I)$ and the BLS low priority is the lowest one. This may not be the case, especially when multiple BLS are considered. As a consequence, the shaper blocking effect may be overestimated, depending on the maximum frame sizes.

The second limitation concerns the shaper blocking as a whole. The definition of both t_I^{R+} and t_I^{S-} are completely independent from the lower priority traffic rates and bursts. As a consequence, if the replenishment intervals are too large in comparison to the traffic load, the shaper blocking is again overestimated: when no lower priority traffic is available, the BLS flows can be sent no matter the state of the credit. The BLS is actually a non-blocking shaper: only the state of the queues and their respective priorities matter.

Finally, the third limitation is due to the blocking shaper computation hypothesis: the busy period considered concerns the class I, but also all lower priority classes. But, we will show in the next section that this can lead to optimistic bounds.

4.2.2 Impact of busy periods

To assess the CPA model optimism, we consider herein the 3-classes case study presented in Section 3.4.1, where the SCT class is shaped by a BLS.

Usually, to compute the worst-case delay using CPA, the main assumption is to consider all the traffics are backlogged. In the case of the BLS however, we will show that this may lead to optimistic bounds.

To compute the worst-case delay, we first detail the case where all classes are backlogged. The resulting credit evolution and the SCT output traffic are visible in Fig.4.1 in plain line (1).

Then, we consider the case where RC traffic is not backlogged between two times $t0$ and $t1$ (dotted lines (2) in Fig.4.1):

- the credit starts at L_M at ti and decreases until it reaches $\frac{L_M}{2}$ at $t0$;
- then it increases until $t1$ when the credit reaches L_M;
- finally it decreases until reaching L_R at $t2$.

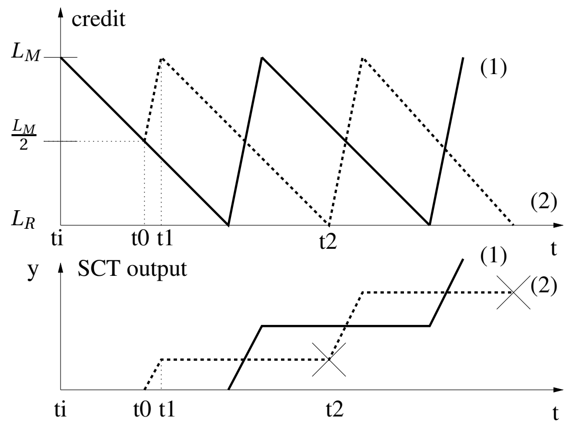


Figure 4.1: Two examples of worst-case BLS behaviour

We see in Fig.4.1 that in this particular case, the SCT output corresponding to the dotted line (2) can be below the one corresponding to the plain line (1). **This shows that the most intuitive worst-case SCT output, i.e., all traffic are backlogged, is not actually the worst-case SCT output.**

From both scenarios presented in Fig.4.1, we have computed in Appendix 8.1 two ***Achievable Worst-Case*** delay bounds for SCT. In the next part, we show the optimism of the CPA model in reference to these achievable worst-case delays.

4.2.3 Discussion

To support our claim, we have implemented the CPA model for three classes as detailed in Appendix 8.2. It is worth noting that for SCT, the implementation is very straightforward: no ILP problem and no fixed-point problem need to be solved. However, this is not the case for RC delay computation, since it requires solving an ILP problem for each defined fixed-point problem when searching for the maximum delay bound.

The first achievable worst-case (AWC-1) has been computed using the same "all traffic are backlogged" hypothesis as CPA and as a consequence both have similar results. The second achievable worst-case (AWC-2) however was computed using a different hypothesis (as illustrated by dotted line (2) in Fig.4.2) and so we use AWC-2 to show the optimism of SCT delay bounds computed with CPA.

We have evaluated the SCT delay bounds with the CPA model for the case study described in Section 3.4.1 under the following parameters:

$$Scenario_{LM} = (UR_{SCT} = 20, UR_{RC} = 20, L_M \in [1382.4..216,830], L_R = 1177.6, BW = 0.46)$$

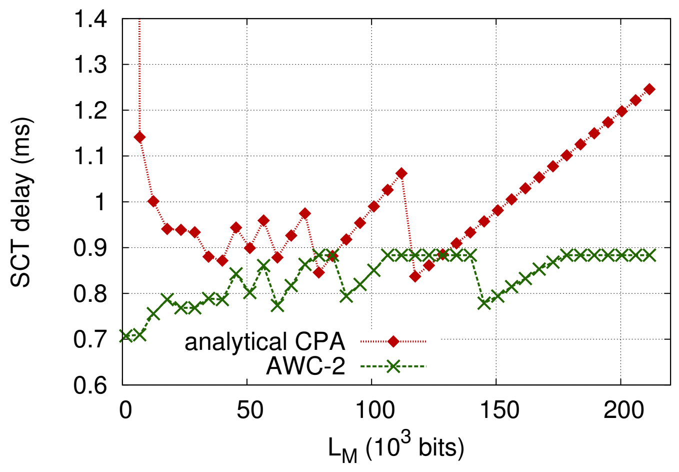


Figure 4.2: CPA model - impact of L_M on SCT delay bounds with $(UR_{SCT} = 20\%, UR_{RC} = 20\%, L_M \in [1382.4..216,830], L_R = 1177.6, BW = 0.46)$

The results are illustrated in Fig.4.2 and it can be noticed that there are several points where the CPA delay bounds are below the achievable worst-case delays, proving that the CPA model can be optimistic. This is because the CPA model considers that all the traffic classes are backlogged during the busy period of SCT to compute the worst-case delay bounds.

On the contrary, for extreme L_M values, the CPA delay bounds are pessimistic compared to AWC-2. To conclude, the CPA model can lead to either pessimistic or optimistic delay bounds, depending on the BLS parameters and input traffics.

To overcome the identified limitations of the CPA modelisation, our proposal is based on the Network Calculus [87] framework. Contrary to CPA, Network Calculus necessitates only the calculation of a maximum and has been proved as highly modular and scalable, in comparison with CPA method [88]. Several existing works have used Network Calculus to analyse the timing performance of Ethernet networks [49] [89] [90]. In particular, Network Calculus has been used to certify the AFDX [27]. To the best of our knowledge, the issue of modelling and analysing the TSN/BLS on top of a NP-SP scheduler using the Network Calculus has not been handled yet in the literature.

4.3 Computing a novel NC model for TSN/BLS

In this section, we start by presenting the followed timing analysis methodology based on the Network calculus framework. Then, we will explain our extended AFDX network modelisation and particularly the BLS impact through the definition of arrival and service curves. Finally, we will discuss the nature of the BLS from the theoretical point of view, in reference to commom "greedy shapers".

4.3.1 Timing analysis methodology

To explain our followed methodology, we first present the Network Calculus framework and we describe the schedulability conditions. Afterwards, we explain how to compute end-to-end delay bounds.

4.3.1.1 Network Calculus framework

The timing analysis used here is based on Network Calculus theory [87] providing upper bounds on delays and backlogs. Delay bounds depend on the traffic arrival described by the so called *arrival curve* α, and on the availability of the traversed node described by the so called minimum *service curve* β. The definitions of these curves are explained as following.

Definition 1 (Arrival Curve)**.** *[87] A function $\alpha(t)$ is an arrival curve for a data flow with an input cumulative function $R(t)$,i.e., the number of bits received until time t, iff:*

$$\forall t, R(t) \leq R \otimes {}^{1} \alpha(t)$$

Definition 2 (Strict minimum service curve)**.** *[87] The function β is the minimum* strict *service curve for a data flow with an output cumulative function R^*, if for any backlogged period $]s,t]^2$, $R^*(t) - R^*(s) \geq \beta(t-s)$.*

Definition 3 (Maximum service curve)**.** *[87] The function $\gamma(t)$ is the maximum service curve for a data flow with an input cumulative function $R(t)$ and output cumulative function $R^*(t)$ iff:*

$$\forall t, R^*(t) \leq R \otimes \gamma(t)$$

To compute end-to-end delay bounds of individual traffic flows, we need the following Theorem.

Theorem 1. *(Blind Multiplex of two flows) [91] Consider two flows f_1,f_2 crossing a system n with the strict minimum service $\beta(t)$, and with the flows f_j α_j-constrained, $j \in \{1,2\}$. Then, the residual minimum service curve offered to f_1 is:*

$$\beta_1^n(t) = (\beta(t) - \alpha_2(t))_\uparrow$$

Then, to compute the main performance metrics, we need the following results.

Corollary 1. *(Left-over service curve - NP-SP Multiplex)[91] Consider a system with the strict service $\beta(t)$ and m flows crossing it, f_1,f_2,..,f_m. The maximum frame size of f_i is MFS_i, its priority is $p(i)$, and f_i is α_i-constrained. The flows are scheduled by the NP-SP policy. For each $i \in \{2,..,m\}$, the strict service curve offered to f_i is given by*[3]*:*

$$\beta_i(t) = \left(\beta(t) - \sum_{\forall j, p(j)<p(i)} \alpha_j(t) - \max_{\forall l, p(l) \geqslant p(i)} MFS_l \right)_\uparrow$$

[1] $f \otimes g(t) = \inf_{0 \leq s \leq t}\{f(t-s) + g(s)\}$

[2] $]s,t]$ is called backlogged period if $R(\tau) - R^*(\tau) > 0, \forall \tau \in]s,t]$

[3] $g_\uparrow(t) = \max\{0, \sup_{0 \leqslant s \leqslant t} g(s)\}$, and $\forall i$, the priorities strictly higher than $p(i)$, are $\forall j, p(j) < p(i)$

Theorem 2 (Performance Bounds). *[87] Consider a flow F constrained by an arrival curve α crossing a system $\mathscr{S}$ that offers a minimum service curve β and a maximum service curve γ. The performance bounds obtained at any time t are:*
Backlog[4]*:* $\forall\ t:\ q(t) \leq v(\alpha,\beta)$
Delay[5]*:* $\forall\ t:\ d(t) \leq h(\alpha,\beta)$
Output arrival curve : $\alpha^*(t) = \alpha \oslash^{6} \beta(t)$
Tight Output arrival curve: $\alpha^*(t) = \big((\gamma \otimes \alpha) \oslash \beta\big)(t)$

The computation of these bounds is greatly simplified in the case of leaky bucket arrival curve $\alpha(t) = b + rt$, with b the maximal burst and r the maximum rate, i.e., the flow is (b,r)-constrained; and the Rate-Latency service curve $\beta_{R,T}(t) = [R\cdot(t-T)]^+$ ($[x]^+$ is the maximum between x and 0) with latency T and rate R. In this case, the delay is bounded by $h(\alpha,\beta) = \frac{b}{R} + T$, and the backlog bound is $v(\alpha,\beta) = b + r\cdot T$. Moreover, the output arrival curve is $\alpha^*(t) = b + r(t+T)$.

In the case of a piecewise linear input arrival curve and a piecewise linear minimum service curve, we can compute the delay bound as follows:

Corollary 2 (Maximum delay bound under a piecewise arrival curve and piecewise minimum service curve). *Consider a flow f constrained by a piecewise linear arrival curve α such as: $\alpha_f(t) = \min_i(\alpha_{r_i,b_i}(t))$, with $\alpha_{r_i,b_i}(t) = r_i\cdot t + b_i$, $i \in [1,n]$ and minimum service curve such as: $\beta_f(t) = \max_j\big(\beta_{R_j,T_j}(t)\big)$, with $\beta_{R_j,T_j}(t) = R_j\cdot(t-T_j)^+$. The maximum latency of flow f is:*

$$delay_f^{max} = \min_j\Big(\frac{y_k}{R_j} + T_j - x_k\Big), \text{with: } k = \min\{i|r_i \leqslant R_j\}$$

and:

$$\begin{cases} x_1 = 0, y_1 = b_1 \\ x_k = \frac{b_k - b_{k-1}}{r_{k-1}-r_k}, y_k = b_k + r_k\cdot x_k, \text{for } 2 \leqslant k \leqslant n \\ x_{n+1} = +\infty, y_{n+1} = +\infty \end{cases}$$

Proof. From Theorem 2, we know that the maximum delay bound of flow f is the maximal horizontal distance between $\alpha_f(t)$ and $\beta_f(t)$. Moreover, from Lemma 1 in [92], we know that the maximum horizontal distance between $\alpha_f(t) = \min_i(\alpha_{r_i,b_i}(t))$ and $\beta_{R_j,T_j}(t) = R_j\cdot(t-T_j)^+$ is:

$$delay^{max} = \frac{y_k}{R_j} + T_j - x_k, \text{with: } k = \min\{i|r_i \leq R_j\}$$

$$\text{and with: } \begin{cases} x_1 = 0, y_1 = b_1 \\ \alpha_k(x_1) = \alpha_{k+1}(x_1) = y_k, \text{for } 1 \leqslant k \leqslant n \\ x_{n+1} = +\infty, y_{n+1} = +\infty \end{cases}$$

[4]v: maximal vertical distance
[5]h: maximal horizontal distance
[6]$f \oslash g(t) = \sup_{s\geq 0}\{f(t+s) - g(s)\}$

With a few algebraic considerations, we deduce that $x_k = \frac{b_k - b_{k-1}}{r_{k-1} - r_k}$, $y_k = b_k + r_k \cdot \frac{b_k - b_{k-1}}{r_{k-1} - r_k}$, for $2 \leqslant k \leqslant n$

Finally, we consider a service curve $\beta_f(t) = \max_j(\beta_{R_j,T_j}(t))$. From Network Calculus concepts in [87], we know that $delay_f^{max} = \inf_{t \geqslant 0}\left\{(\alpha_f \oslash \beta_f)(-t) \leqslant 0\right\}$. Thus, if we consider a piecewise service curve we have:

$$\begin{aligned}
(\alpha \oslash \beta)(t) &= \sup_{s \geqslant 0}\left\{\alpha(t+s) - \beta(s)\right\} \\
&= \sup_{s \geqslant 0}\left\{\alpha(t+s) - \max_j(\beta_{R_j,T_j}(s))\right\} \\
&= \sup_{s \geqslant 0}\left\{\min_j\left[\alpha(t+s) - \beta_{R_j,T_j}(s)\right]\right\} \\
&= \min_j\left[\sup_{s \geqslant 0}\left\{\alpha(t+s) - \beta_{R_j,T_j}(s)\right\}\right]
\end{aligned}$$

Hence, to compute the maximum delay we can compute the maximum delay for every $\beta_{R_j,T_j}(t)$ and keep the minimum value. □

Theorem 3 (Concatenation-Pay Bursts Only Once)**.** *[87] Assume a flow crossing two servers with respective service curves β_1 and β_2. The system consisting of the concatenation of the two servers offers a service curve $\beta_1 \otimes \beta_2$.*

4.3.1.2 Schedulability Conditions

To infer the real-time guarantees of our proposed solution, we need first to define a **necessary schedulability condition**. This consists in respecting the stability condition within the network, where the sum of maximum arrival rates of the input traffic flows i at any crossed node n has to be lower than its minimum guaranteed service rate within the node n . This constraint is denoted as ***rate constraint***:

$$\forall \text{node } n \in \text{ network}, \sum_{\forall i \ni n} r_i \leqslant R_n$$

Then, we define a **sufficient schedulability condition** to infer the traffic schedulability, which consists in comparing the upper bound on end-to-end delay of each traffic flow f of a class k to its deadline, denoted $Deadline_f^{end2end}$. This constraint is called ***deadline constraint***:

$$\forall \text{class } k, \forall \text{flow } f \in k, delay_{k,f}^{end2end} \leqslant Deadline_{k,f}^{end2end}$$

For this sufficient schedulability condition, we detail the end-to-end delay expression of a flow f in the class k, $delay_{k,f}^{end2end}$, along its path $path_f$ as follows:

$$delay_{k,f}^{end2end} = delay_{k,f}^{es} + delay_{k,f}^{prop} + \sum_{sw \in path_f} delay_{k,f}^{sw} \tag{4.1}$$

with $delay^{es}_{k,f}$ the delay within the source end-system es to transmit the flow f of class k and $delay^{prop}_{k,f}$ the propagation delay along the path, which is generally negligible in an avionics network.

The last delay $delay^{sw}_{k,f}$ represents the upper bound of **the delay within each intermediate switch** along the flow path, and it consists of several parts as shown in Fig.4.3:

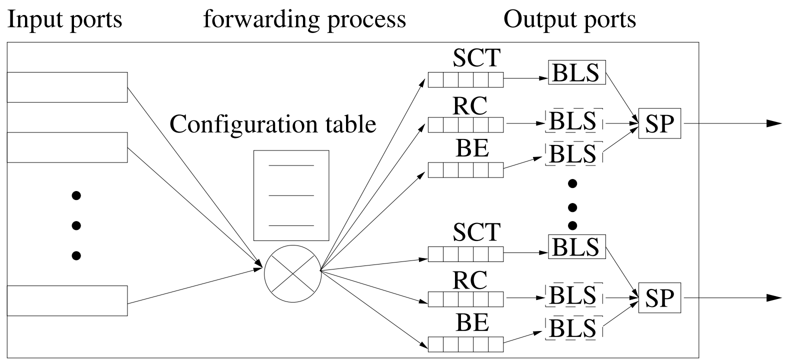


Figure 4.3: An extended AFDX switch architecture with 3 classes

- **the store and forward delay** at the input port, equal to $\frac{MFS}{C}$, with MFS the length of the frame and C the capacity;
- **the technological latency** due to the forwarding process, upper-bounded by 1μs in the pre-specification of the AFDX next generation;
- **the output port multiplexer delay** due to the BLS and NP-SP scheduler, denoted $delay^{mux}_{k,f}$. Hence, the only two unknown are the delays in the end-system and the output port of the switch. To enable the computation of upper bounds on these delays, we need to model the different parts of the network, and more particularly the BLS.

4.3.1.3 Computing End-to-End Delay Bounds

The computation of the end-to-end delay upper bounds follows four main steps:

1. computing the strict minimum service curve guaranteed to each traffic class k in each node n of type $\{es, mux\}$, β^n_k, will infer the computation of the residual service curve, guaranteed to each individual flow f of class k, $\beta^n_{k,f}$ with Theorem 1;
2. knowing the residual service curve guaranteed to each flow within each crossed node allows the propagation of the arrival curves along the flow path, using Theorem 2. We can compute the output arrival curve, based on the input arrival curve and the minimum service curve, which will be in its turn the input of the next node;
3. the computation of the minimum end-to-end service curve of each flow f in class k, based on Theorem 3, is simply the concatenation of the residual service curves, $\beta^n_{k,f}$, $\forall n$ along its path $path_f$;

4. given the minimum end-to-end service curve of each flow f in class k along its $path_f$ and its maximum arrival curve at the initial source, the end-to-end delay upper bound $delay_{k,f}^{end2end}$ is the maximum horizontal distance between the two curves using Theorem 2 and Corollary 2.

Hence, as we can notice, we need to model all the unknown service curves, in the end-systems and in the switch output ports to enable the end-to-end delay upper bounds computation. These curves are detailed in the next section. It is worth noting that since the BE class has no deadline, the service curves guaranteed to this class and the computation of the respective upper bounds on end-to-end delays are not detailed here, but can easily be computed using Corollary 1 and Theorem 2.

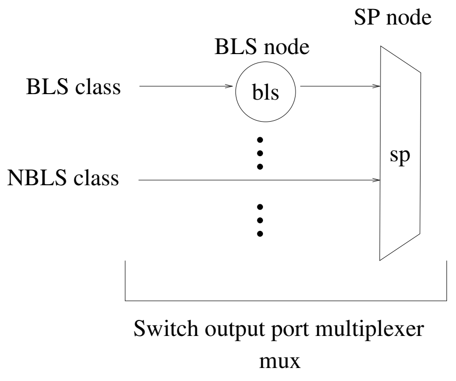


Figure 4.4: Output port multiplexer node nomenclature

In the next Section, we present the modelisation of the extended AFDX network starting with the traffic and end-system modelling and continuing with the BLS model. As illustrated in Fig.4.4, an output port mux consist of a NP-SP node sp and a BLS node bls. We denote BLS classes the classes shaped by a BLS, and $NBLS$ classes the classes not shaped by a BLS. Hence, the modelisation of the BLS node bls is necessary to compute the service curve of output port mux, $\beta_k^{mux}(t)$, $\forall$ class $k \in \{BLS, NBLS\}$.

4.3.2 Extended AFDX network modelisation

We can now model the different parts of our extended AFDX: traffic, the end-systems es, the switch output port multiplexers mux and particularly the BLS node bls. We start by presenting the considered traffic model.

4.3.2.1 Traffic Modelling

To compute upper bounds on end-to-end delays of different traffic classes using Network Calculus, we need to model each message flow to compute its maximum arrival curve.

The arrival curve of each flow f in class k at the input of the node $n \in \{es, sw\}$ or a component $n \in \{bls, sp\}$ along its path is a leaky-bucket curve with a burst $b_{k,f}^n$ and a rate $r_{k,f}^n$:

$$\alpha_{k,f}^n(t) = b_{k,f}^n + r_{k,f}^n \cdot t$$

Therefore, the arrival curve of the aggregate traffic in class k at the input (resp. output) of the node $n \in \{es, sw\}$ or a component $n \in \{bls, sp\}$ is: $\alpha_k^n(t) = \sum_{f \in k} \alpha_{k,f}^n(t)$ (resp. $\alpha_k^{*,n}(t) = \sum_{f \in k} \alpha_{k,f}^{*,n}(t)$ based on Theorem 2).

Each traffic flow f of class k, generated by an end-system, is characterised by (BAG_f, MFS_f, J_f) for respectively the minimum inter-arrival time, the maximum frame size integrating the protocol overhead, and the jitter.

Hence, the arrival curve of traffic class k in the end-system es, based on a leaky bucket model, is as follows:

$$\begin{aligned}\alpha_k^{es}(t) &= \sum_{f \in k} \alpha_{k,f}^{es}(t) = \sum_{f \in k} MFS_f + \frac{MFS_f}{BAG_f}\left(t + J_f\right) \\ &= b_k + r_k t \text{ with } \begin{cases} b_k = \sum\limits_{f \in k} MFS_f + \frac{MFS_f}{BAG_f} J_f \\ r_k = \sum\limits_{f \in k} \frac{MFS_f}{BAG_f} \end{cases}\end{aligned}$$

4.3.2.2 End-System Modelling

For the end-systems, they are implementing a NP-SP scheduler. This scheduler has been already modelled in the literature [91] through Corollary 1, and the defined strict minimum service curve guaranteed to a traffic class $k \in \{SCT, RC, BE\}$ within an end-system es is as follows:

$$\beta_k^{es}(t) = \left[C \cdot t - \sum_{\forall (i,f), f \in i, p(i) < p(k)} \alpha_{i,f}^{es}(t) - \max_{\forall (i,f), f \in i, p(i) \geqslant p(k)} MFS_f \right]_{\uparrow}$$

4.3.2.3 BLS node model: Window-based Approach (WbA)

In this section, we describe our proposed BLS node model.To compute the guaranteed service curves by the BLS to a class k, we need to detail two types of windows, which are enforced by the BLS behaviour. The first one is denoted as *sending window*, during which the class k has the high BLS priority and is sent uninterruptedly until the consumed credit reaches the maximum threshold, L_M^k. The second one is called *idle window* where the class k has the low BLS priority and the consumed credit is decreasing uninterruptedly until reaching the minimum threshold, L_R^k. Moreover, due to the non-preemptive message transmission, both windows have minimal and maximal durations, as illustrated in Fig.4.5.

For each class k shaped by a BLS, with a BLS high priority $p_H(k)$ and a BLS low priority $p_L(k)$, (with $p_L(k) > p_H(k)$ and priority 0 the highest priority), we define:

- MC(k) the set of classes with a priority strictly between the low BLS priority $p_L(k)$ and the high BLS priority $p_H(k)$, i.e., $\forall j$ such as: $p_H(k) < p(j) < p_L(k)$;
- LC(k) the set of classes with a priority strictly lower than $p_L(k)$, i.e., $\forall j$ such as: $p_L(k) < p(j)$;

- HC(k) the set of classes with a priority strictly higher than $p_H(k)$, i.e., $\forall j$ such as: $p_H(k) > p(j)$.

Strict Minimum Service curve

The strict minimum service curve of a BLS class k defines a lower bound on the class-k output cumulative traffic from the BLS. This curve represents the most deteriorated behaviour of BLS, in terms of offered service to class k, which maximises its delay within the BLS.

Hence, to cover this worst-case behaviour, we combine the maximum *idle window* and the minimum *sending window* durations illustrated in Fig.4.5.

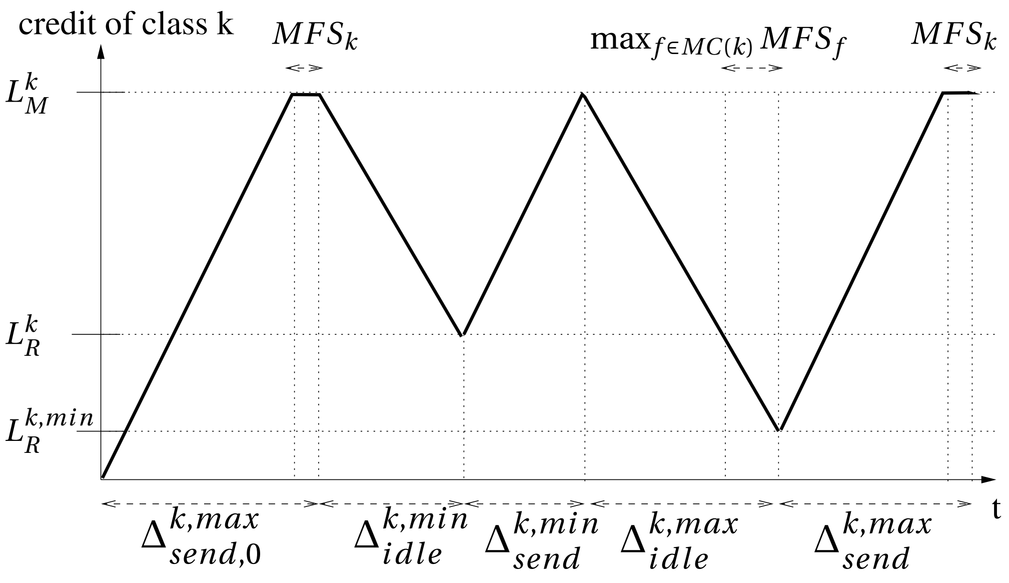


Figure 4.5: Idle and sending windows of a class k

To compute idle and sending windows, we consider only the impact of MC(k) and class k. The impact of HC(k) flows is taken into account when considering the residual service curve offered to class k. The LC(k) flows have a priority strictly lower than class-k BLS low priority. Thus, the $LC(k)$ frames can never be sent during idle windows.

The minimum *sending window* duration $\Delta_{send}^{k,min}$, illustrated in Fig.4.5, is the time for the consumed credit to go from the lowest to the highest thresholds, i.e., from L_R^k to L_M^k, with an increasing slope I_{send}^k:

$$\Delta_{send}^{k,min} = \frac{L_M^k - L_R^k}{I_{send}^k} \tag{4.2}$$

The maximum *idle window* duration $\Delta_{idle}^{k,max}$, illustrated in Fig.4.5, is the time for the consumed credit to go from L_M^k to L_R^k with a decreasing slope I_{idle}^k, in addition to the transmission time of a maximum frame of the RC traffic. The latter is due to the non-preemption feature when a MC(k) frame is starting its transmission just before the consumed credit reaches the lowest threshold, L_R^k.

$$\Delta_{idle}^{k,max} = \frac{L_M^k - L_R^k}{I_{idle}^k} + \frac{\max_{f \in MC(k)} MFS_f}{C} \tag{4.3}$$

Therefore, the strict minimum service curve guaranteed to the class-k, β_k^{bls}, is defined in Theorem 4.

Theorem 4 (Strict Minimum Service Curve offered to a BLS class k by a BLS node)**.** *Consider a server with a constant rate C, implementing BLS shapers. The traffic class k crosses this server and is shaped by the BLS. The input arrival curve of a flow $j \in HC(k)$ in the NP-SP node is $\alpha_j(t) = r_j \cdot t + b_j$.*

The strict minimum service curve guaranteed to the BLS class k is as follows:

$$\beta_{k \in BLS}^{bls}(t) = \frac{\beta(\Delta_{send}^{k,min})}{\beta(\Delta_{send}^{k,min}) + \beta(\Delta_{idle}^{k,max})} \cdot \left(C - \sum_{j \in HC(k)} r_j\right) \cdot \left(t - \Delta_{idle}^{k,max}\right)^+ \tag{4.4}$$

with:

$$\beta(t) = \left(C - \sum_{j \in HC(k)} r_j\right) \cdot t - \sum_{j \in HC(k)} b_j$$

where $[x]^+$ is the maximum between x and 0.

Proof. We present here a sketch of proof. The full proof is detailed in Appendix 8.3.1. Using the credit variations due to the sending and idle windows, we are able to compute the lower bound for the output cumulative function of class k, which is also the strict minimum service curve. We obtain the curve illustrated in Fig.4.6 in the particular case of the 3-classes case study presented in Section 3.4.1.

□

Corollary 3 (Strict Minimum Service Curve offered to SCT by a BLS node in the case of three traffic classes)**.** *Consider a server with a constant rate C, implementing a BLS shaping the SCT traffic. The SCT, RC and BE traffics cross this server with the following priorities: $p(SCT) \in \{p_H(SCT) = 0, p_L(SCT) = 2\}$, $p(RC) = 1$, $p(BE) = 3$, as illustrated in Fig.3.3.*

The strict minimum service curve guaranteed to SCT is as follows:

$$\beta_{SCT}^{bls}(t) = \frac{\Delta_{send}^{min}}{\Delta_{send}^{min} + \Delta_{idle}^{max}} \cdot C \cdot \left(t - \Delta_{idle}^{max}\right)^+$$

with $\Delta_{send}^{min} = \frac{L_M - L_R}{I_{send}}$ and $\Delta_{idle}^{max} = \frac{L_M - L_R}{I_{idle}} + \frac{MFS_{RC}}{C}$

Proof. We apply Theorem 4 in the particular 3-classes case study presented in Fig.3.1, with SCT as class k. Thus $HC(k) = \emptyset$, MC(k)=RC, and $\beta(t) = C \cdot t$. □

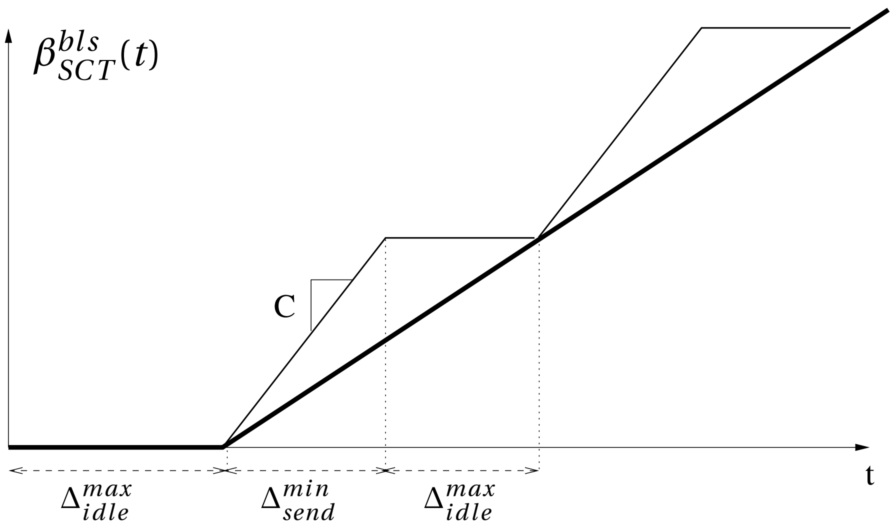


Figure 4.6: Strict minimum service curve $\beta_{SCT}^{bls}(t)$

Maximum Service curve

The maximum service curve of BLS class k represents the best offered service to class k, which induces the minimum processing delay within the BLS. As such, in the presence of MC(k) traffic, we combine the minimum *idle window* duration and the maximum *sending window* one to handle this best-case behaviour.

We consider the best-case scenario, when neither HC(k) not LC(k) interfere with class-k traffic. So as before, we focus on the impact of MC(k) and class k on idle and sending windows.

The maximum *sending window* duration $\Delta_{send}^{k,max}$, illustrated in Fig.4.5, is equal to the sum of:

- the time necessary for the gained credit to go from L_R^k to L_M^k with an increasing slope I_{send}^k, equals to $\frac{L_M^k - L_R^k}{I_{send}^k}$;
- the transmission time of a maximum-sized class-k frame due to the non-preemption feature, i.e., one class-k frame may start its transmission just before the consumed credit reaches L_M^k, equals to $\frac{MFS_k}{C}$;
- the time to consume the gained credit during the transmission of one additional maximum frame of MC(k) traffic at the end of the *idle window*. The latter parameter is due to the fact that the resume level of BLS, L_R^k, is the lower threshold on the consumed credit to trigger the priority change of the class k from low to high, but not an extreme value for the consumed credit itself (L_M^k or 0).

  So, if a frame of MC(k) traffic has been transmitted just at the end of the *idle window*, the consumed credit keeps decreasing until it either reaches 0, or the transmission ends.

Therefore, the lowest value the consumed credit can reach due to the non-preemption feature illustrated in Fig.4.5 is:

$$L_R^{k,min} = \max(0, L_R^k - \frac{\max_{f\in MC(k)} MFS_f}{C}.I_{idle}^k)$$

The additional time during which the consumed credit can then increase with a slope I_{send}^k is $\frac{L_R^k - L_R^{k,min}}{I_{send}^k} = \min(\frac{\max_{f\in MC(k)} MFS_f}{C} \cdot \frac{I_{idle}^k}{I_{send}^k}, \frac{L_R^k}{I_{send}^k})$.

Thus, the maximum *sending window* duration is as follows:

$$\Delta_{send}^{k,max} = \frac{L_M^k - L_R^k}{I_{send}^k} + \frac{MFS_k}{C} + \min(\frac{\max_{f\in MC(k)} MFS_f}{C} \cdot \frac{I_{idle}^k}{I_{send}^k}, \frac{L_R^k}{I_{send}^k}) \tag{4.5}$$

However, it is worth noting that the consumed credit may start at 0, such as at the initialisation phase or after a long period of inactivity. Hence, the maximum initial *sending window* duration, denoted $\Delta_{send,0}^{k,max}$ and illustrated in Fig.4.5 covers such possibility, and is as follows:

$$\Delta_{send,0}^{k,max} = \frac{L_M^k}{I_{send}^k} + \frac{MFS_k}{C} \tag{4.6}$$

The minimum *idle window* duration $\Delta_{idle}^{k,min}$, illustrated in Fig.4.5, is simply the time it takes for the consumed credit to go from L_M^k to L_R^k with a decreasing slope of I_{idle}^k:

$$\Delta_{idle}^{k,min} = \frac{L_M^k - L_R^k}{I_{idle}^k} \tag{4.7}$$

Therefore, the maximum service curve guaranteed to the BLS class k, $\gamma_k^{bls}(t)$, is defined in Theorem 5.

Theorem 5 (Maximum Service Curve offered to a BLS class k by a BLS node)**.** *Consider a server with a constant rate C, implementing BLS shapers. The traffic class k crosses this server and is shaped by the BLS.*

The maximum service curve guaranteed to the class-k traffic is as follows.

$$\gamma_{k\in BLS}^{bls}(t) = \begin{cases} \text{if MC(k) traffic is enqueued: } \frac{C}{\Delta\gamma_k} \cdot \Delta_{send}^{k,max} \cdot t + \frac{C}{\Delta\gamma_k} \cdot \Delta_{send,0}^{k,max} \cdot \Delta_{idle}^{k,min} \\ \text{Otherwise: } C \cdot t \end{cases}$$

with $\Delta\gamma_k = \Delta_{send}^{k,max} + \Delta_{idle}^{k,min}$.

Proof. We present here a sketch of proof. The full proof is detailed in Appendix 8.3.2. In the general case, the maximum service curve is $C \cdot t$. When MC(k) is enqueued, we compute a tighter maximum service curve. The main idea is very similar to the proof in Appendix 8.3.1. Using the relation linking sending and idle windows through the credit variations, we are able to compute the upper bound for the output cumulative function, which is also the maximum service curve. We obtain the leaky-bucket curve illustrated in Fig.4.7 in the particular case of the 3-classes case study presented in Section 3.4.1.

□

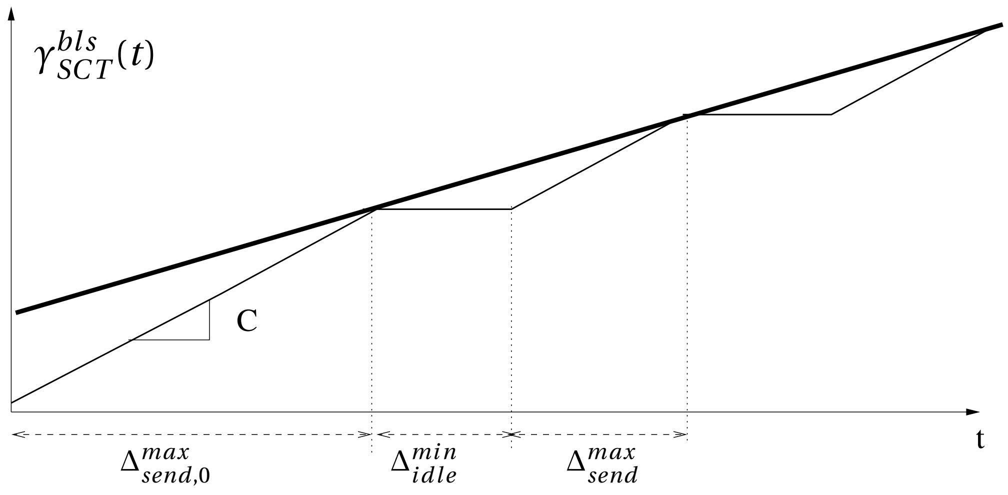


Figure 4.7: Maximum service curve $\gamma^{bls}_{SCT}(t)$ when MC(k) traffic is enqueued

Corollary 4 (Maximum Service Curve offered to SCT by a BLS node in the case of three traffic classes)**.** *Consider a server with a constant rate C, implementing a BLS shaping the SCT traffic. The SCT, RC and BE traffics cross this server with the following priorities:* $p(SCT) \in \{p_H(SCT) = 0, p_L(SCT) = 2\}$, $p(RC) = 1$, $p(BE) = 3$, *as illustrated in Fig.3.3.*

The maximum service curve guaranteed to the SCT traffic is as follows.

$$\gamma^{bls}_{SCT}(t) = \begin{cases} \textit{if RC traffic is enqueued: } \frac{C}{\Delta\gamma_{SCT}} \cdot \Delta^{max}_{send} \cdot t + \frac{C}{\Delta\gamma_{SCT}} \cdot \Delta^{max}_{send,0} \cdot \Delta^{min}_{idle} \\ \textit{Otherwise: } C \cdot t \end{cases}$$

with $\Delta\gamma_{SCT} = \Delta^{max}_{send} + \Delta^{min}_{idle}$, $\Delta^{max}_{send} = \frac{L_M - L_R}{I_{send}} + \frac{MFS_{SCT}}{C} + \min(\frac{MFS_{RC}}{C} \cdot \frac{I_{idle}}{I_{send}}, \frac{L_R}{I_{send}})$ *and* $\Delta^{min}_{idle} = \frac{L_M - L_R}{I_{idle}}$

Proof. We apply Theorem 5 to the particular case of the 3-classes case study presented in Fig.3.1, with SCT as class k and RC as MC(k). □

Maximum Output Arrival curve

The maximum output arrival curve of a BLS class k is detailed in the following Corollary:

Corollary 5 (Maximum Output Arrival Curve of a BLS class)**.** *Consider a BLS class k with a maximum leaky-bucket arrival curve α at the input of a BLS shaper, guaranteeing a minimum rate-latency service curve* β^{bls}_k *and a maximum service curve* γ^{bls}_k. *The maximum output arrival curve is:*

$$\alpha^{*,bls}_k(t) = \min(\gamma^{bls}_k(t), \alpha \oslash \beta^{bls}_k(t))$$

Proof. To prove Corollary 5, we generalise herein the rule 13 in p. 123 in [87], i.e., $(f \otimes g) \oslash g \leq f \otimes (g \oslash g)$, to the case of three functions f, g and h when $g \oslash h \in \mathscr{F}$, where $\mathscr{F}$ is the set of non negative and wide sense increasing functions:

$$\mathscr{F} = \{f : \mathbb{R}^+ \to \mathbb{R}^+ \mid f(0) = 0, \forall t \geq s : f(t) \geq f(s)\}$$

According to Theorem 2, we have $\alpha^*(t) = (\gamma_k^{bls} \otimes \alpha) \oslash \beta_k^{bls}$. Moreover, in the particular case of a leaky-bucket arrival curve α and a rate-latency service curve β_k^{bls}, $\alpha \oslash \beta_k^{bls}$ is a leaky-bucket curve, which is in $\mathscr{F}$. Hence, we have the necessary condition to prove the following:

$$(\alpha \otimes \gamma) \oslash \beta(t) \leq \gamma \otimes (\alpha \oslash \beta)(t) \leq \min(\gamma(t), \alpha \oslash \beta(t))$$

$$\begin{aligned}
&(\alpha \otimes \gamma) \oslash \beta(t) \\
&= \sup_{u \geq 0} \left\{ (\gamma \otimes \alpha)(t+u) - \beta(u) \right\} \\
&= \sup_{u \geq 0} \left\{ \inf_{-u \leq s' \leq t} \left\{ \gamma(t-s') + \alpha(s'+u) - \beta(u) \right\} \right\} \\
&\leq \sup_{u \geq 0} \left\{ \inf_{0 \leq s' \leq t} \left\{ \gamma(t-s') + \alpha(s'+u) - \beta(u) \right\} \right\} \\
&\leq \sup_{u \geq 0} \left\{ \inf_{0 \leq s' \leq t} \left\{ \gamma(t-s') + \sup_{v \geq 0} \left\{ \alpha(s'+v) - \beta(v) \right\} \right\} \right\} \\
&= \gamma \otimes (\alpha \oslash \beta)(t) \leq \min(\gamma(t), \alpha \oslash \beta(t))
\end{aligned}$$

□

4.3.2.4 Output port multiplexer modelisation

In this section, we compute the strict minimum service curves offered by a switch output port multiplexer mux. Such a multiplexer mux consists of sp and bls nodes as illustrated in Fig.4.4.

The strict minimum service curve offered by each output port multiplexer to a BLS class is defined in Theorem 6, and the strict minimum service curves offered by each output port multiplexer to a $NBLS$ class, i.e., not shaped by a BLS, is defined in Theorem 7.

Theorem 6 (Strict Minimum Service Curve offered by an output port multiplexer to a BLS class). *Consider a system implementing a BLS with the strict minimum service β.*

The strict minimum service curve offered to a BLS class k by an output port multiplexer is:

$$\beta_{k \in BLS}^{mux}(t) = \max\left(\beta_{k \in BLS, p_L(k)}^{sp}, \beta_{k \in BLS}^{bls} \otimes \beta_{k \in BLS, p_H(k)}^{sp}\right)(t)$$

with:

- $\beta_{k \in BLS, p_L(k)}^{sp}(t) = (\beta(t) - \sum_{j \in MC(k) \cup HC(k)} \alpha_j^{sp}(t) - \max_{j \in LC(k) \cup k} MFS_j)_\uparrow$ *the strict minimum service curve offered by the NP-SP when the class-k BLS priority is low;*

- $\beta_{k \in BLS}^{bls}(t)$ *the strict minimum service curve offered by the BLS node to class k, defined in Theorem 4;*

- $\beta_{k \in BLS, p_H(k)}^{sp}(t) = (\beta(t) - \sum_{j \in HC(k)} \alpha_j^{sp}(t) - \max_{j \notin HC(k)} MFS_j)_\uparrow$ *the strict minimum service curve offered by the NP-SP when the class-k BLS priority is high;*

- $\alpha_j^{sp}(t)$ *the input arrival curve of flow j at the sp node, such as:*

$$\begin{cases} \textit{if } j \in NBLS, \alpha_j^{sp}(t) = \alpha_j(t) \\ \textit{if } j \in BLS, \alpha_j^{sp}(t) = \alpha_j^{*,bls}(t) = \min(\gamma_j^{bls}, \alpha_j^{bls} \oslash \beta_j^{bls})(t), \textit{defined in Corollary 5} \end{cases}$$

Proof. The idea is to model the impact of a BLS implemented on top of the NP-SP scheduler on BLS class k. To achieve this aim, we distinguish two possible scenarios. The first one covers the particular case where the class-k priority remains low, i.e., the other queues are empty; whereas the second one covers the general case where the priority of class k oscillates between $p_L(k)$ and $p_H(k)$, as explained in Section 3.2.1. Firstly, the minimum service curve guaranteed within mux in the first scenario is due to the NP-SP scheduler and denoted $\beta^{sp}_{k\in BLS,p_L(k)}$, which is computed via Corollary 1 when considering the class-k priority is $p_L(k)$. Secondly, the minimum service curve guaranteed within mux in the second scenario is computed via Theorem 3, through the concatenation of the service curves within the BLS node $\beta^{bls}_{k\in BLS}$ (computed in Theorem 4) and the NP-SP node $\beta^{sp}_{k\in BLS,p_H(k)}$ (computed via Corollary 1 when class-k priority is high). □

Corollary 6 (Strict Minimum Service Curve offered by an output port multiplexer to SCT in the case of three traffic classes)**.** *Consider a server with a constant rate C, implementing a BLS shaping the SCT traffic. The SCT, RC and BE traffics cross this server with the following priorities: $p(SCT) \in \{p_H(SCT) = 0, p_L(SCT) = 2\}$, $p(RC) = 1$, $p(BE) = 3$, as illustrated in Fig.3.3.*

The strict minimum service curve offered to SCT by an output port multiplexer is:

$$\beta^{mux}_{SCT}(t) = \max\left(\beta^{sp}_{SCT,2}, \beta^{bls}_{SCT} \otimes \beta^{sp}_{SCT,0}\right)(t)$$

with:

- $\beta^{sp}_{SCT,2}(t) = (C \cdot t - \alpha_{RC}(t) - \max_{j\in BE\cup SCT} MFS_j)_\uparrow$ *the strict minimum service curve offered by the NP-SP when the class-k BLS priority is low;*

- $\beta^{bls}_{SCT}(t)$ *the strict minimum service curve offered by the BLS node to SCT, defined in Corollary 3;*

- $\beta^{sp}_{SCT,0}(t) = (C \cdot t - \max_{j\in\{SCT,RC,BE\}} MFS_j)_\uparrow$ *the strict minimum service curve offered by the NP-SP when the BLS priority is high;*

Proof. We apply Theorem 6 in the particular 3-classes case study presented in Fig.3.1, with SCT as class k. Thus $HC(k) = \emptyset$, MC(k)=RC, and $\beta(t) = C \cdot t$. □

Theorem 7 (Strict Minimum Service Curve offered to a NBLS class by an output port multiplexer)**.** *Consider a system implementing a BLS with the strict service β and m flows crossing it, f_1,f_2,...,f_m. The strict minimum service curve offered to a NBLS class k by an output port multiplexer is:*

$$\beta^{mux}_{k\in NBLS}(t) = \max\left(\beta^{sp}_{k\in NBLS}, \beta^{bls}_{k\in NBLS}\right)(t)$$

with:

- $\beta^{sp}_{k\in NBLS} = (\beta - \sum_{p_H(j)<p(k), j\in BLS} \alpha_j \oslash \beta^{bls}_j - \sum_{p(j)<p(k), j\in NBLS} \alpha_j - \max_{p(j)\geqslant p(k)} MFS_j)_\uparrow$;
- β^{bls}_j*, with $j \in BLS$, the strict minimum service curve offered by the BLS node to class j, defined in Theorem 4;*
- $\beta^{bls}_{k\in NBLS} = (\beta - \sum_{p_H(j)<p(k), j\in BLS} \gamma^{bls}_j - \sum_{p(j)<p(k), j\in NBLS} \alpha_j - \max_{p(j)\geqslant p(k)} MFS_j)_\uparrow$;
- γ^{bls}_j*, with $j \in BLS$, the maximum service curve offered by the BLS node to class j, defined in Theorem 5.*

Proof. The proof of Theorem 7 is straightforward. Theorem 7 is obtained through replacing within the equation of Corollary 1 the arrival curve of higher priority traffic than class $k \in NBLS$ by the curves computed in Corollary 5. □

Corollary 7 (Strict Minimum Service Curve offered to RC by an output port multiplexer in the case of three traffic classes)**.** *Consider a server with a constant rate C, implementing a BLS shaping the SCT traffic. The SCT, RC and BE traffics cross this server with the following priorities: $p(SCT) \in \{p_H(SCT) = 0, p_L(SCT) = 2\}$, $p(RC) = 1$, $p(BE) = 3$, as illustrated in Fig.3.3.*

The strict minimum service curve offered to RC by an output port multiplexer is:

$$\beta^{mux}_{RC}(t) = \max\left(\beta^{sp}_{RC}, \beta^{bls}_{RC}\right)(t)$$

with:

- $\beta^{sp}_{RC}(t) = (C\cdot t - \alpha_{SCT}(t) \oslash \beta^{bls}_{SCT}(t) - \max_{j\in\{SCT,RC,BE\}} MFS_j)_\uparrow$;
- $\beta^{bls}_{SCT}(t)$ *the strict minimum service curve offered by the BLS node to SCT, defined in Corollary 3;*
- $\beta^{bls}_{RC}(t) = (C\cdot t - \gamma^{bls}_{SCT}(t) - \max_{j\in\{SCT,RC,BE\}} MFS_j)_\uparrow$;
- $\gamma^{bls}_{SCT}(t)$ *the maximum service curve offered by the BLS node to SCT, defined in Corollary 4.*

Proof. We apply Theorem 7 to the particular case of the 3-classes case study presented in Fig.3.1, with SCT as class k and RC as MC(k). □

Now that we have modelised the proposed network, we use this model to answer the question whether "shaper" is the correct qualifier for the BLS.

4.3.3 Discussion: is the BLS really a shaper?

The most common kind of shapers is the greedy shaper, which has been detailed in [87]. According to [87], a *shaper* with a shaping curve σ is a bit processing device that forces its output to have σ as an output arrival curve. A *greedy shaper* is a shaper that delays the input bits in a buffer, whenever sending a bit would violate the constraint σ, but outputs them as soon as possible. A consequence of this definition is that, for an input flow R, the output flow R^* is defined by $R^* = R \otimes \sigma$. Moreover, as the service curve β and maximum service curve γ are defined by $R^* \geqslant R \otimes \beta$ and $R^* \leqslant R \otimes \gamma$, this means that $\sigma = \beta = \gamma$ in the case of a greedy shaper. Obviously, this is not the case for the BLS. Another property of the greedy shaper is that the difference between the fluid model and the packetized model is bounded by the maximum sized packet.

From the BLS WbA model in Section 4.3.2.3, we can easily compute the corresponding fluid (bit-per-bit) WbA model: we do not consider an additional frame due to non-preemption. As a consequence, the defined windows are $\Delta^{k,min}_{send}$, $\Delta^{k,min}_{idle}$,and $\Delta^{k,min}_{send,0} = \frac{L^k_M}{I^k_{send}}$.

After simple calculations, we obtain that $\frac{\Delta^{k,min}_{send}}{\Delta^{k,min}_{send}+\Delta^{k,min}_{idle}} = I^k_{idle}$ and $\frac{\Delta^{k,min}_{idle}}{\Delta^{k,min}_{send}+\Delta^{k,min}_{idle}} = I^k_{send}$. So, when considering the 3-classes case study presented in Section 3.4.1, we have:

$$\gamma^{bls,fluid}_{SCT}(t) = \frac{\Delta^{min}_{send}}{\Delta^{min}_{send}+\Delta^{min}_{idle}} \cdot C \cdot t + \Delta^{min}_{send,0} \cdot C \cdot \frac{\Delta^{min}_{idle}}{\Delta^{min}_{send}+\Delta^{min}_{idle}} = I_{idle} \cdot t + L_M = \gamma^{bls,intui,fluid}_{SCT}(t)$$

$$\beta^{bls,fluid}_{SCT}(t) = \frac{\Delta^{min}_{send}}{\Delta^{min}_{send}+\Delta^{min}_{idle}} \cdot C \cdot \left(t - \Delta^{min}_{idle}\right)^+ = I_{idle} \cdot \left(t - \frac{L_M - L_R}{I_{idle}}\right)^+ = \beta^{bls,intui,fluid}_{SCT}(t)$$

where $\beta^{bls,intui,fluid}_{SCT}(t)$ and $\gamma^{bls,intui,fluid}_{SCT}(t)$ are the intuitive fluid models of minimum and maximum service curves defined in Appendix 8.4 in Eq.(8.12) and Eq.(8.14), respectively. Hence, we obtain an interesting insight about the behaviour of the BLS during the worst-case.

This shows that the difference between the fluid and packetized models, i.e., $\beta^{bls,fluid}_k$ and β^{bls}_k, is larger than a single maximum sized frame: an additional MC(k) frame is considered in every idle window, and an additional frame of class k is considered in each sending window

Finally, the BLS functioning itself shows that the BLS is not a greedy shaper: if a frame is enqueued and there is no higher priority frame enqueued, then the frame is dequeued no matter the state of the BLS credit. Hence, the BLS is non-blocking contrary to the definition of a greedy shaper. Moreover, if no higher priority traffic is present, then the BLS does not force the output to conform to a certain σ, unlike a shaper.

So, if the BLS is not a shaper, what is its nature? The BLS changes the priority of a queue through reordering the priority of the different queues, and it cannot be used without a Static Priority Scheduler. So trying to characterise it on its own is futile. Together with the NP-SP however, they are able to reorganise the output traffic according to the BLS parameters. Because of this, BLS+SP is much closer to schedulers such as Deficit Round Robin (DRR) than shapers.

4.4 Preliminary performance evaluation

In this section, we start by evaluating the tightness and sensitivity of our model, in reference to Achievable Worst-Cases (AWCs) described in Appendix 8.1. Next, we compare the CPA and NC models under different scenarios. We finish by comparing the extended AFDX incorporating BLS, the current AFDX implementing NP-SP and an AFDX incorporating DRR.

4.4.1 Case study

We consider the case study presented in Section 3.4.1, a single-hop Gigabit network described in Fig.3.7.

To evaluate our model, we conduct tightness and sensitivity analyses using different scenarios when varying the input rates of SCT and RC and the BLS parameters. The first two scenarios are identical to those in Chapter 3, and the third is the one used in Section 4.2.3 to prove the CPA model optimism. The five scenarios are described by the following vectors:

$$Scenario_{SCT} = (UR_{SCT} \in [0.1:78], UR_{RC} = 20, L_M = 22,118, L_R = 0, BW = 0.46)$$

$$Scenario_{RC} = (UR_{SCT} = 20, UR_{RC} \in [0.5:72], L_M = 22,118, L_R = 0, BW = 0.46)$$

$$Scenario_{LM} = (UR_{SCT} = 20, UR_{RC} = 20, L_M \in [1382.4..216,830], L_R = 1177.6, BW = 0.46)$$

$$Scenario_{LR} = (UR_{SCT} = 20, UR_{RC} = 20, L_M = 22,118, L_R \in [0..0.99] \cdot L_M, BW = 0.46)$$

$$Scenario_{BW} = (UR_{SCT} = 20, UR_{RC} = 20, L_M = 22,118, L_R = 1177.6, BW \in [0..0.99])$$

Finally, since there is no strict order between the two achievable worst-cases (see Appendix 8.1), we will use the maximum value, denoted AWC=max(AWC-1, AWC-2), as a reference to assess our model tightness.

4.4.2 Sensitivity and Tightness Analyses

In this section, we analyse our model by assessing first, the impact of the BLS parameters and the utilisation rates of SCT and RC on the delay bounds, then its tightness in reference to AWC.

4.4.2.1 Sensitivity Analysis

In this section, we analyse the sensitivity of the BLS model when varying the BLS parameters and utilisation rates, i.e., UR_{SCT}, UR_{RC}, L_M, L_R, BW. The results of the different scenarios are reported in Fig.4.8, Fig.4.9 Fig.4.10, Fig.4.11, and Fig.4.12.

From our modelisation of the output port multiplexer and the BLS node, we notice that in $\beta_{SCT}^{mux}(t)$ (see Corollary 6) and $\beta_{RC}^{mux}(t)$ (see Corollary 7) the evolution of the strict minimum service curves of SCT and RC follows the maximum of two linear curves: one is the SP part, the other is the BLS part. Consequently, the delay bounds of SCT and RC evolve also following two parts under the different scenarios.

Impact of UR_{SCT}

In Fig.4.8(a), when the SCT utilisation rate increases, we observe an increase of the SCT delay bounds starting close to 0 thanks to a low initial latency and high rate of the guaranteed minimum service curve due to the BLS part. Then, at $UR_{SCT} = 20\%$, the delay bounds due to the BLS part reaches the delay bounds due to the SP part. After this point, the delay follows the maximum rate according to Corollary 6, i.e., the SP part. Furthermore, in Fig.4.8(b), when the SCT utilisation rate increases, we observe a noticeable increase of the RC delay bounds following the increasing guaranteed rate of the service curve due to the SP part. Then, after $UR_{SCT} = 18\%$, the delay bounds becomes constant since it depends on the strict minimum service curve due to the BLS part, which is constant when the RC utilisation rate is constant.

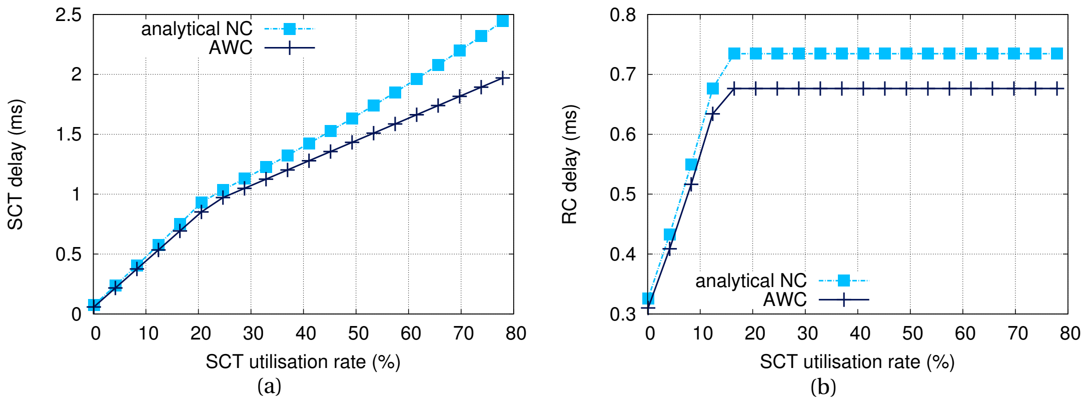


Figure 4.8: NC vs AWC - impact of SCT maximum utilisation rate on: (a) SCT delay bounds; (b) RC delay bounds, with $Scenario_{SCT} = (UR_{SCT} \in [0.1:78], UR_{RC} = 20, L_M = 22,118, L_R = 0, BW = 0.46)$

Hence, this analysis shows that the SCT utilisation rate has an inherent impact on SCT and RC delay bounds, where:

- the SCT delay bound is ruled below $UR_{SCT} = 20\%$ by the strict minimum service curve due to the BLS part, $(\beta^{sp}_{SCT,0} \otimes \beta^{bls}_{SCT})(t)$; whereas after $UR_{SCT} = 20\%$, it is ruled by the strict minimum service curve due to the SP part, $\beta^{sp}_{SCT,2}(t)$;
- the RC delay bound is ruled below $UR_{SCT} = 18\%$ by the strict minimum service curve due to the SP part, $\beta^{sp}_{RC}(t)$; whereas after $UR_{SCT} = 18\%$, it is ruled by the strict minimum service curve due to the BLS part, $\beta^{bls}_{RC}(t)$.

These results infer that the variation of UR_{SCT} has a large impact on both the SCT (resp. RC) delay bounds with a maximum variation of 2.5ms (resp. 0.4ms), i.e., the delay bound is multiplied by 24 (resp. 2.3).

Impact of UR_{RC}

Similar analysis conducted for $Scenario_{RC} = (UR_{SCT} = 20, UR_{RC} \in [0.5:72],$ $L_M = 22,118, L_R = 0, BW = 0.46)$ in Fig.4.9 shows that the RC utilisation rate has an inherent impact on SCT and RC delay bounds. We observe a behaviour symmetrical to the one noticed in $Scenario_{SCT}$:

- the SCT delay bound is ruled below $UR_{RC} = 20\%$ by the strict minimum service curve due to the SP part; whereas after $UR_{RC} = 20\%$, it is ruled by the strict minimum service curve due to the BLS part;
- the RC delay bound is ruled below $UR_{RC} = 30\%$ by the strict minimum service curve the BLS part; whereas after $UR_{RC} = 30\%$, it is ruled by the strict minimum service curve the SP part.

These results infer that the variation of UR_{RC} has a large impact on both the SCT (resp. RC) delay bounds with a maximum variation of 0.58ms (resp. 2.2ms), i.e., the delay bound is multiplied by 2.8 (resp. 23).

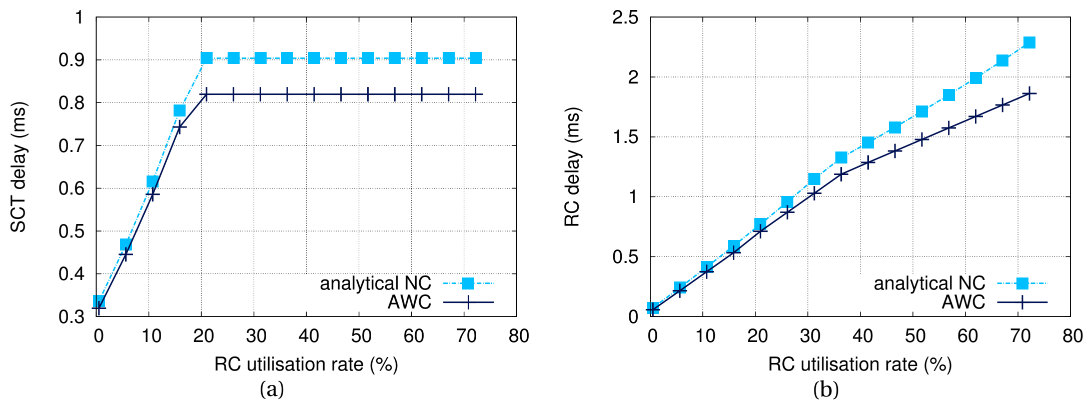


Figure 4.9: NC vs AWC - impact of RC maximum utilisation rate on: (a) SCT delay bounds; (b) RC delay bounds, with $Scenario_{RC} = (UR_{SCT} = 20, UR_{RC} \in [0.5:72], L_M = 22,118, L_R = 0, BW = 0.46)$

Finally, it is worth noting that our model has the same behaviour for SCT and RC (two linear curves) as the simulations in Fig.3.8 and Fig.3.9 from Chapter 3.

Impact of L_M

Concerning SCT delay bounds, in Fig.4.10(a), before $L_M = 50,000$ bits (which represents sending windows allowing the transmission of 200 consecutive frames) they are ruled by β^{bls}_{SCT}. The increase of L_M increases both idle and sending windows Δ^{max}_{idle} and Δ^{min}_{send}. Thus, both the rate and the initial latency of the minimum service curve β^{bls}_{SCT} (see Corollary 6) increase. This fact induces a variation of the SCT delay bounds, which decrease then increase. After $L_M = 50,000$ bits, the SCT delay bound is constant, because it is ruled by a constant $\beta^{sp}_{SCT,0}$.

Concerning RC delay bounds, in Fig.4.10(b), they are ruled by $\beta_{RC}^{bls}(t)$ (see Corollary 7). When increasing L_M, both the rate and initial latency increase. Before $L_M = 7,000$ bits (which represents sending windows allowing the transmission of 24 consecutive frames), the impact of the increasing rate is stronger, resulting in the delay bound decrease; whereas after $L_M = 7,000$ bits, the impact of the initial latency takes over, resulting in a delay bound increase.

These results infer that the variation of L_M has a limited impact on the SCT delay bounds with a maximum variation of 5%, and a larger effect on RC delay bounds with a variation of 35%.

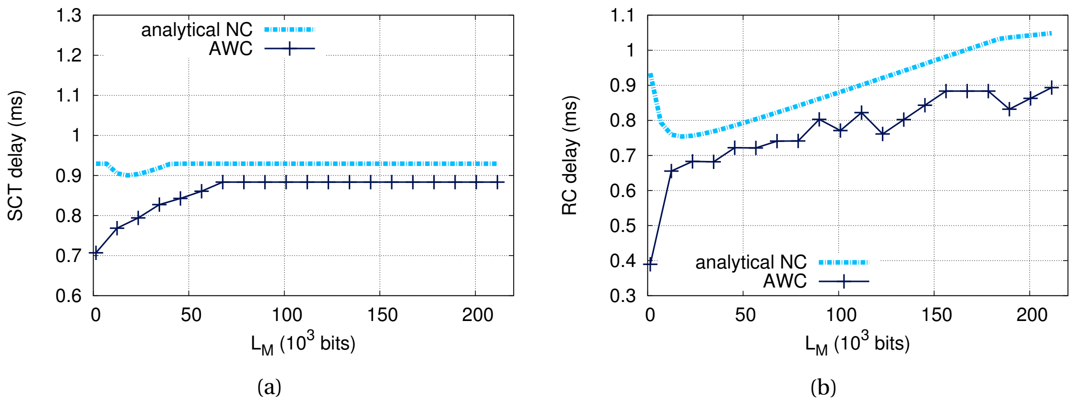


Figure 4.10: NC vs AWC - impact of L_M on: (a) SCT delay bounds; (b) RC delay bounds, with $Scenario_{LM} = (UR_{SCT} = 20, UR_{RC} = 20, L_M \in [1382.4..216,830], L_R = 1177.6, BW = 0.46)$

Impact of BW

As shown in Fig.4.11(a), when BW is below 40%, the SCT delay bound is constant since it is ruled by $\beta_{SCT,2}^{sp}$ (see Corollary 6): the bandwidth allocated by the BLS is not sufficient to send the SCT traffic. As a consequence, the SCT traffic also uses the bandwidth left by the RC traffic. However, for BW higher than 40%, SCT delay bound decreases. This is due to the fact that $I_{idle} = BW \cdot C$, thus the guaranteed rate of the SCT minimum service curve β_{SCT}^{bls} increases while its initial latency decreases.

Concerning the RC delay bounds, we observe in Fig.4.11 the opposite behaviours: for BW higher than 55%, RC delay bound is constant and ruled by β_{RC}^{sp} (see Corollary 7); whereas for BW lower than 55%, RC delay bound is ruled by β_{RC}^{bls}. Thus, when increasing BW, I_{idle} increases. This leads to decreasing the guaranteed rate of β_{RC}^{bls}, and consequently to the delay bounds increase.

These results infer that the variation of BW has a high impact on both SCT and RC delay bounds with an increase of 0.60ms (resp. 0.55ms) for SCT (resp. RC), representing an increase of 170% (resp. 137%).

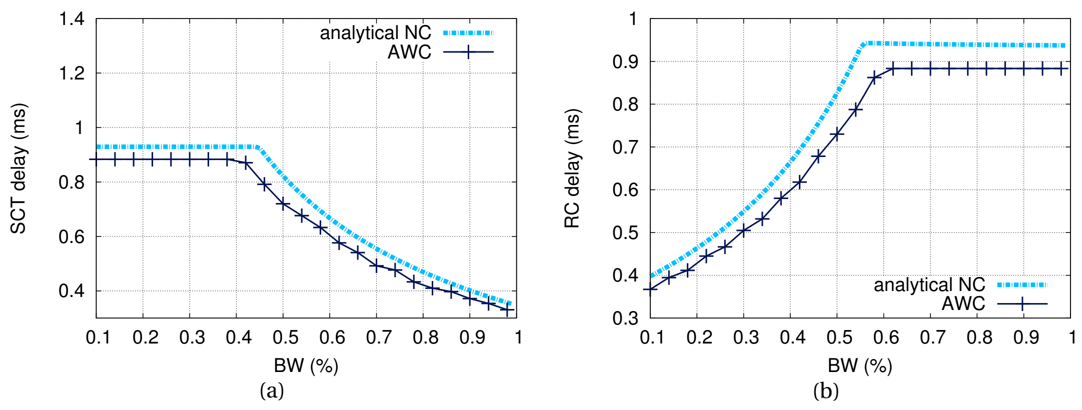


Figure 4.11: NC vs AWC - impact of BW on: (a) SCT delay bounds; (b) RC delay bounds, with $Scenario_{BW} = (UR_{SCT} = 20, UR_{RC} = 20, L_M = 22,118, L_R = 1177.6, BW \in [0..0.99])$

Impact of L_R

In Fig.4.12(a), we see that increasing L_R from 40% to 70% slightly increases SCT delay bounds. Before $L_R = 70\%$, the equation ruling the SCT behaviour is $\beta^{bls}_{SCT} \otimes \beta^{sp}_{SCT,0}$ (see Corollary 6). The increase of L_R decreases both Δ^{max}_{send} and Δ^{min}_{idle} (see Eq.(4.5) and Eq.(4.7)), leading here to a low increase of SCT delay bounds. After 70%, the SCT behaviour is ruled by $\beta^{sp}_{SCT,2}$ (see Corollary 6). Since $\beta^{sp}_{SCT,2}$ does not depend on BLS parameters, the SCT delay bound becomes constant after $L_R = 70\%$.

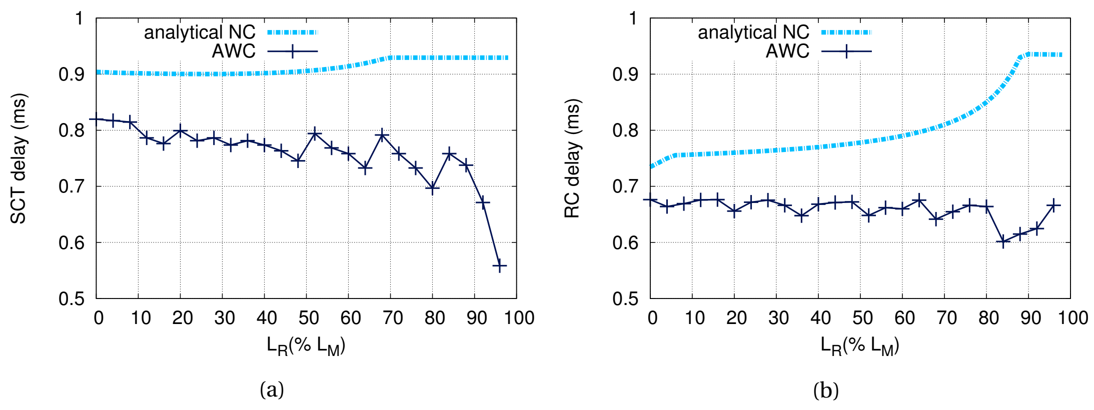


Figure 4.12: NC vs AWC - impact of L_R on: (a) SCT delay bounds; (b) RC delay bounds, with $Scenario_{LR} = (UR_{SCT} = 20, UR_{RC} = 20, L_M = 22,118, L_R \in [0..0.99] \cdot L_M, BW = 0.46)$

In Fig.4.12(b), the RC delay bound is ruled by β^{bls}_{RC}, until $L_R = 90\%$, after which it becomes ruled by β^{sp}_{RC}. Hence, when L_R increases, $L_M - L_R$ decreases, leading to a decrease of the minimum service rate and consequently an increase of the RC delay bound.

These results show that L_R has a limited impact on the SCT delay with a maximum variation of 2%; whereas its impact is higher on RC delay bounds, with a variation of 26%.

To conclude the sensitivity analysis of the WbA model, both L_R and L_M have very limited impact on the SCT delay bound (under 5%), but a large one on RC delay bounds (around 30%). Moreover, the largest impact however, is due to the the SCT and RC utilisation rates and BW parameter, with delay bound increases up to 137% and 170% for RC and SCT respectively.

4.4.2.2 Tightness Analysis

Thanks to the modelisation of both the BLS and SP parts of the output port multiplexer, the SCT and RC delay bounds are very tight in reference to AWC, under the different considered scenarios. For instance, as shown in Fig.4.8(a), when varying the SCT utilisation rate, the maximum gap between the AWC and the NC delay bounds of SCT is $0.5ms$, which represents an increase of 33%. Moreover, when varying the RC utilisation rate, we have similar results: the largest percentage increase of the RC delay bounds happens for a gap of $0.5ms$ and represents 27% in Fig.4.9(b). When varying BW, we also have similar results in Fig.4.11 for both SCT and RC delay bounds.

However, when the BLS parameters come close to open limits, we can see that the WbA model is less tight. For instance, when L_R is close to L_M in Fig.4.10 and Fig.4.12, the largest gap for SCT delay bounds compared to AWC is 0.25ms, which represents an increase up to 36%. This is because $L_M = L_R$ is a forbidden state as it results in a null minimum service rate in β^{bls}_{SCT} and β^{bls}_{RC}. Hence, when the guaranteed rate is close to 0, the delay bounds increase, until reaching the limits set by $\beta^{sp}_{SCT,0}$ and β^{sp}_{RC}.

Finally, with a gap between AWC and the NC model usually below 15% under the various scenarios, the proposed model can be considered as an accurate one.

Thus, the tightness of the model is very high: the gap between the NC model and AWC is usually less than 15%, with peaks at 35%. However, we note a situation where the tightness of the model could be improved: when L_M is close to L_R.

4.4.3 Comparing CPA and NC models

In this section, we compare our proposed model to the CPA model. We start by comparing the computation times, before studying the SCT and RC delay bounds.

4.4.3.1 Computation times

In this section, we consider the computation time to obtain all the computed times under each scenario.

With our NC model, we compute each delay bound through simple linear computations. With CPA however, the computation is much more complex:

- SCT delay bounds necessitate finding a maximum using a while-loop;

- RC delay bounds necessitate solving:
  - a maximisation problem;
  - fixed point problems;
  - ILP problems.

We can see clearly in Table 4.1 that the NC model necessitates much less computation power than the CPA model. In fact, we can notice that the NC model is between 20,000 and 100,000 times faster than the CPA model.

scenario	CPA (s)	NC (s)	CPA/NC
varying SCT	97.2	0.0051	19,058
varying RC	71.4	0.0032	22,187
varying L_M	384.4	0.0072	53,388
varying L_R	1059	0.0100	105,900
varying BW	390	0.0095	41,052

Table 4.1: CPA and NC models computation times for the different scenarios

4.4.3.2 SCT delay bounds

We can see in Fig.4.13(a) that the SCT delay bounds of the two models are overlapping for low values of UR_{SCT}. Then, they diverge at $UR_{SCT} = 20\%$. The analytical NC curve starts to follow a linear curve with a lower increase rate. The CPA model however, keeps the same rate. As a consequence, the gap between CPA and NC curves increases (up to 70%), which shows the increasing pessimism of CPA under high SCT utilisation rates.

The main cause of this pessimism is due to the fact that the CPA shaper blocking does not take into account the RC rate. As BW is close to 50%, the idle and send slopes are very close: the replenishment and service intervals are very similar. So, when the SCT utilisation rate becomes visibly larger than the RC one (over $UR_{SCT} = 20\%$), the replenishment intervals are not completely filled: SCT traffic is sent even-though the SCT priority is low. This causes the decreasing SCT delay bounds under the NC model. Similar results are visible when varying the different parameters in Fig.4.14(a), Fig.4.15(a), Fig.4.17(a), and Fig.4.16(a), where the delay bounds are generally more pessimistic with CPA model than the ones with NC model.

However, in Fig.4.15(a), we can see that SCT delay bounds with CPA are sometimes lower than the ones with NC model. This fact confirms our conclusions in Section 4.2.1 about the CPA model optimism.

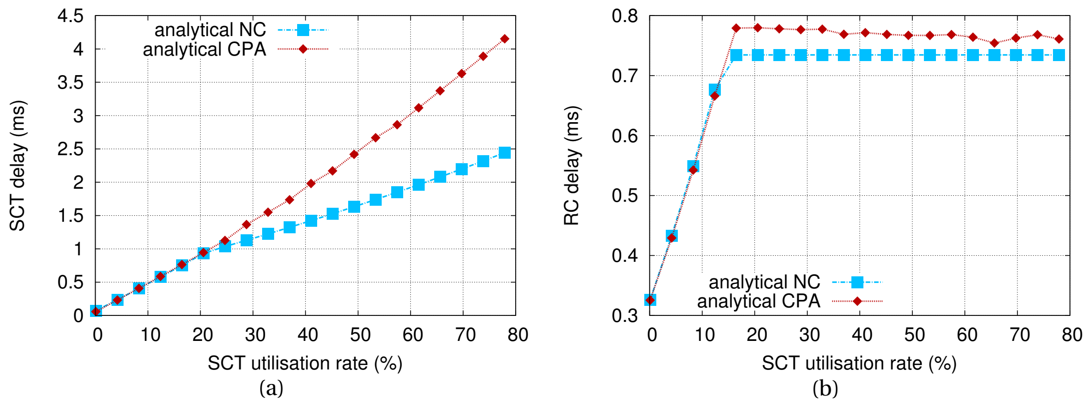


Figure 4.13: NC vs CPA - impact of SCT maximum utilisation rate on: (a) SCT delay bounds; (b) RC delay bounds, with $Scenario_{SCT} = (UR_{SCT} \in [0.1:78], UR_{RC} = 20, L_M = 22,118, L_R = 0, BW = 0.46)$

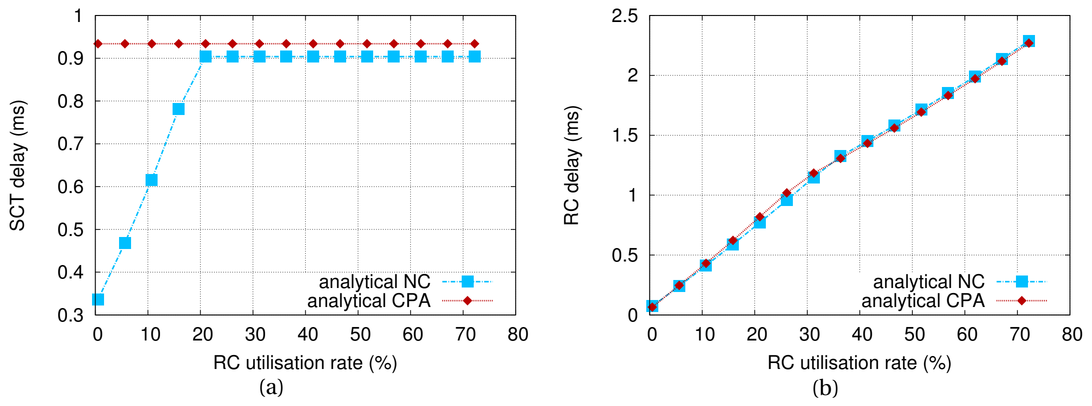
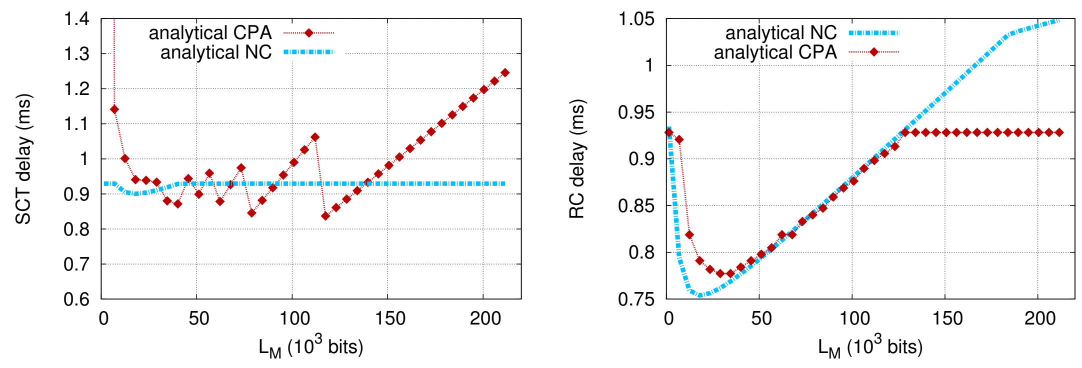


Figure 4.14: NC vs CPA - impact of RC maximum utilisation rate on: (a) SCT delay bounds; (b) RC delay bounds, with $Scenario_{RC} = (UR_{SCT} = 20, UR_{RC} \in [0.5:72], L_M = 22,118, L_R = 0, BW = 0.46)$

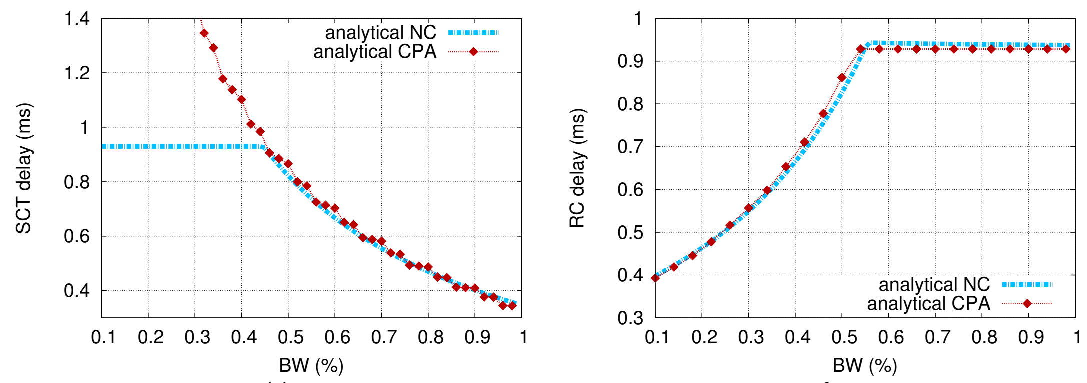


(a) (b)

Figure 4.15: NC vs CPA - impact of L_M on: (a) SCT delay bounds; (b) RC delay bounds, with $Scenario_{LM} = (UR_{SCT} = 20, UR_{RC} = 20, L_M \in [1382.4..216,830], L_R = 1177.6, BW = 0.46)$

4.4.3.3 RC delay bounds

Concerning the RC traffic, in Fig.4.13(b)) and Fig.4.14(b) both analytical models have the same shape. In NC, this is again thanks to the association of the BLS and SP parts, β_{RC}^{bls} and β_{RC}^{sp}. In the CPA model, this is thanks to solving an ILP problem, which takes into account the maximum available SCT traffic.

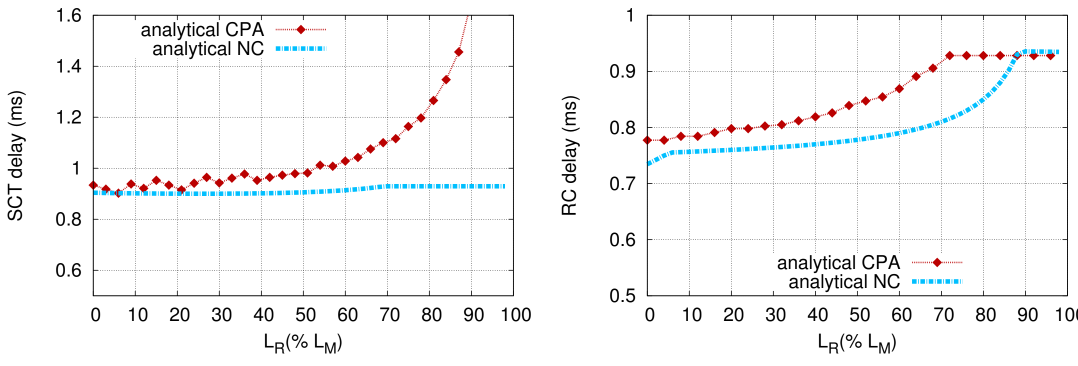


(a) (b)

Figure 4.16: NC vs CPA - impact of BW on: (a) SCT delay bounds; (b) RC delay bounds, with $Scenario_{LR} = (UR_{SCT} = 20, UR_{RC} = 20, L_M = 22,118, L_R \in [0..0.99] \cdot L_M, BW = 0.46)$

It is worth noting that in the part of the curve ruled by the sp node, NC and CPA have very similar delay bounds; whereas in the part ruled by the bls node, CPA leads to slightly larger delay bounds. This can be explained by the pessimism of maximum replenishment intervals t_{SCT}^{R+} in the ILP problem of CPA: the credit replenishment is over-evaluated as explained in Section 4.2.1. Hence, the resulting sending interval is also over-evaluated, which adds pessimism to the RC delay bounds.

An exception of this general behaviour is visible in Fig.4.15(b) where for large L_M, the CPA model is less pessimistic than the NC model (up to 10%).



(a) (b)

Figure 4.17: NC vs CPA - impact of L_R on: (a) SCT delay bounds; (b) RC delay bounds, with $Scenario_{BW} = (UR_{SCT} = 20, UR_{RC} = 20, L_M = 22,118, L_R = 1177.6, BW \in [0..0.99])$

From this analysis, we can point out the low complexity of our model compared to the CPA model. Moreover, concerning the SCT traffic, we have solved the optimism issue of the CPA model, and most of the pessimism issues too. Finally, both CPA and NC models lead to very close RC delay bounds except for large values of either L_M or L_R.

4.4.4 Comparing extended AFDX (BLS), current AFDX (SP), and DRR-compliant AFDX

As presented in the related work, an other possible solution for mixed-criticality applications is the Deficit Round Robin (DRR). Hence, in this part, we will compare the extended AFDX to the current AFDX, i.e., a 3-priority Non-preemptive Static Priority Scheduler, and to a DRR-compliant AFDX, i.e., an AFDX incorporating a DRR scheduler. First, we detail the DRR architecture and parameters in Section 4.4.4.1. Afterwards, we present the comparative analysis of computed SCT and RC delay bounds, based on $Scenario_{SCT}$ and $Scenario_{RC}$.

4.4.4.1 DRR-compliant AFDX architecture and parametrisation

As our aim is to compare the extended AFDX to the DRR, the first challenge is to find a scheduling architecture with the DRR equivalent to the one with the BLS on top of the NP-SP. In our proposed implementation of the extended AFDX incorporating the BLS, shown in Fig.3.1, the shaped queue is the highest priority: SCT. Thus, the BLS shares the bandwidth between the SCT and the other priorities, RC and BE. An equivalent of this is a DRR with two queues, one for the SCT traffic and one for both RC and BE traffic. A second important aspect of the extended AFDX is the association of the BLS to the NP-SP scheduler that privileges the

RC traffic over the BE traffic. To keep this aspect within the DRR-compliant AFDX architecture, we use a NP-SP upstream the DRR to privilege RC over BE, as illustrated in Fig.4.18.

From [45], we know that the minimum service curves offered by the DRR to SCT and RC∪BE in an output port mux are as following:

$$\begin{aligned}\beta_{SCT}^{DRR,mux}(t) &= \frac{Q_{SCT}}{Q_{SCT}+Q_{RC\cup BE}}\cdot C\cdot\Big(t-\frac{Q_{SCT}\cdot(\max_{i\in RC\cup BE}MFS_i-\epsilon)}{Q_{SCT}\cdot C}\\ &\quad+\frac{Q_{RC\cup BE}\cdot(Q_{SCT}+MFS_{SCT}-\epsilon)}{Q_{SCT}\cdot C}\Big)^+\end{aligned}$$

$$\begin{aligned}\beta_{RC\cup BE}^{DRR,mux}(t) &= \frac{Q_{RC\cup BE}}{Q_{RC\cup BE}+Q_{SCT}}\cdot C\cdot\Big(t-\frac{Q_{RC\cup BE}\cdot(MFS_{SCT}-\epsilon)}{Q_{RC\cup BE}\cdot C}\\ &\quad+\frac{Q_{SCT}\cdot(Q_{RC\cup BE}+\max_{i\in RC\cup BE}MFS_i-\epsilon)}{Q_{RC\cup BE}\cdot C}\Big)^+\end{aligned}$$

with Q_{SCT} and Q_{RC} the quantums of SCT and RC respectively, and ϵ is the basic unit, e.g., 1 when considering bytes and 8 when considering bits.

With Theorem 3, we can concatenate the SP and the DRR to obtain the service curve offered to the RC traffic by the DRR-compliant AFDX output port multiplexer:

$$\beta_{RC}^{DRR,mux}(t) = [\beta_{RC\cup BE}^{DRR,mux}(t)-\max_{i\in RC\cup BE}MFS_i]_\uparrow \tag{4.8}$$

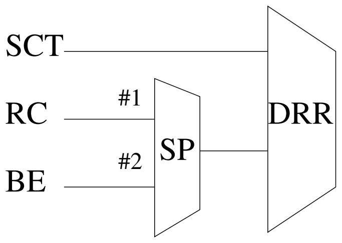


Figure 4.18: DRR-compliant AFDX output port architecture

We can see in Fig.4.8(b) (resp. Fig.4.9(a)) that RC (resp. SCT) delay bound is constant after a certain value of UR_{SCT} (resp. UR_{RC}). Hence, to compare DRR-compliant and extended AFDX, we set DRR weights such as the delay bounds of SCT and RC under DRR-compliant AFDX are similar to the delay bounds with extended AFDX in the constant part, e.g., when UR_{SCT} is over 18%, and when UR_{RC} is over 20%.

First, we set the SCT Quantum Q_{SCT} such as the number of SCT frames sent in a BLS sending window is the same with DRR:

$$Q_{SCT}=\frac{L_M}{(1-BW)\cdot MFS_{SCT}}.$$

Then, we set the RC Quantum Q_{RC} to achieve the constant delay bound under $Scenario_{SCT}$ when UR_{SCT} is over 18% as follows:

$$Q_{RC}=\lfloor(1-BW)\cdot Q_{SCT}\cdot 1.85\rfloor$$

and under $Scenario_{RC}$ when UR_{RC} is over 18%:

$$Q_{RC} = \lfloor(1 - BW) \cdot Q_{SCT} \cdot 2.55\rfloor$$

Now that we have presented the parameterisation of the DRR-compliant AFDX architecture, we can detail the comparative results.

4.4.4.2 Comparative analysis

Impact of SCT

First, in Fig.4.19(a) we can see that the SCT delay bounds increase with both DRR-compliant and extended AFDX (BLS). However, the BLS delay bounds are higher than the DRR delay bounds before UR_{SCT} reaches 32%, and become lower for high UR_{SCT}. Moreover, in Fig.4.19(b), we can see that the RC delay bound is higher with DRR for low SCT utilisation rates (under 15%), compared to the extended AFDX. Over 15%, DRR-compliant and extended AFDX are equal and constant. Afterwards, we can notice that both DRR-compliant and extended AFDX induce a noticeable reduction of the RC delay bounds compared to the current AFDX (SP). For instance for UR_{SCT} = 40%, both the DRR-compliant and extended AFDX divide the RC delay bound by 2.6 compared to the current AFDX. It is interesting to note that while the RC delay bound with DRR-compliant AFDX can be higher than with current AFDX, it is either equal or lower than current AFDX with the extended AFDX. This fact shows the beneficial impact of the BLS on the RC delay bounds, in comparison with DRR.

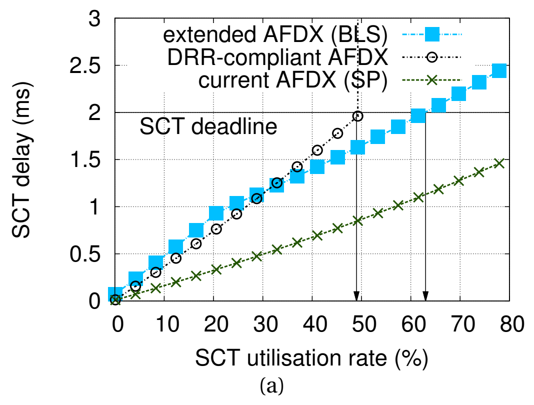
Moreover, we notice that the SCT schedulability is enhanced by the BLS, in comparison to DR and SP. For instance in Fig.4.19(a):

- for current AFDX (SP), the maximum UR_{SCT} is 42%, when the RC deadline is crossed;
- for DRR-compliant AFDX, the maximum UR_{SCT} is 49%, when the SCT deadline is crossed;
- for extended AFDX (BLS), the maximum UR_{SCT} is 63%, when the SCT deadline is crossed;

Hence, the extended AFDX increases the schedulability of the SCT traffic by 50% compared to the current AFDX, and by 28% compared to DRR-compliant AFDX. Additionally, it is interesting to note that above UR_{SCT} = 49%, DRR drops SCT frames, which is not the case for BLS.

Impact of RC

First, in Fig.4.20(a), we can see that as before the SCT delay bound increases with both DRR-compliant and extended AFDX, even-though they remain lower with extended AFDX until UR_{RC} = 20%. Then, both mechanisms lead to equal and constant delay bounds as expected from the DRR parameterisation, explained in Section 4.4.4.1. In Fig.4.20(b), we can see that the RC delay bound is slightly lower with DRR-compliant AFDX than the extended AFDX, until a RC utilisation rate of 45%. Then, DRR-compliant AFDX delay bound becomes the highest. As before, while the RC delay bound with DRR-compliant AFDX can be higher than with the current AFDX, the extended AFDX is either equal or lower than current AFDX. So, the RC delay bound is positively impacted by the BLS, in comparison to DRR. For instance, the delay bound is divided by 2 for UR_{RC} = 10%, in comparison to the current AFDX (SP).



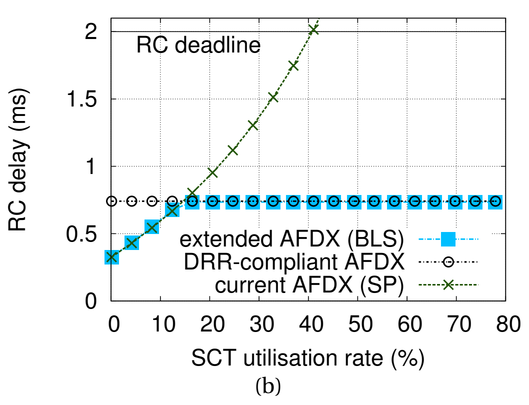
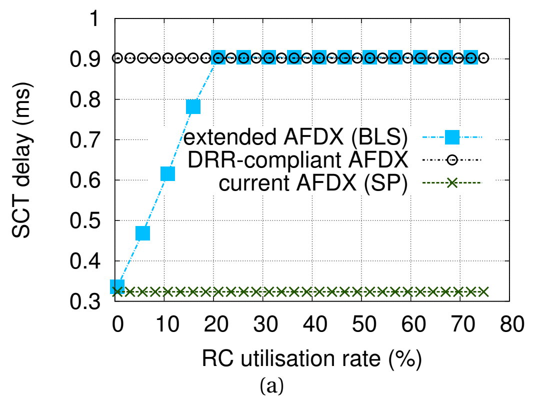


Figure 4.19: BLS vs (SP,DRR) - impact of SCT maximum utilisation rate on: (a) SCT delay bounds; (b) RC delay bounds with $Scenario_{SCT} = (UR_{SCT} \in [0.1:78], UR_{RC} = 20, L_M = 22, 118, L_R = 0, BW = 0.46)$

Concerning the schedulability, the deadlines are still fulfilled for different values of UR_{RC}. For instance in Fig.4.20(b), for DRR-compliant AFDX, the maximum UR_{RC} is 57% and for current and extended AFDX, the maximum UR_{RC} is 62%.

Hence, the BLS increases the schedulability of the SCT traffic by 8.7%, compared to DRR. Moreover, there are DRR drops, this time of RC frames, above 57%.



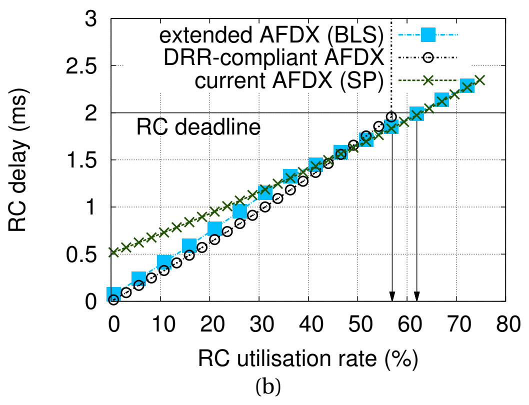


Figure 4.20: BLS vs (SP,DRR) - impact of RC maximum utilisation rate on: (a) SCT delay bounds; (b) RC delay bounds with $Scenario_{RC} = (UR_{SCT} = 20, UR_{RC} \in [0.5:72], L_M = 22, 118, L_R = 0, BW = 0.46)$

Finally, we can conclude that the BLS largely improves the schedulability of SCT compared to both DRR and NP-SP. Moreover, both DRR-compliant and extended AFDX can largely decrease the RC delay bounds. Depending on the utilisation rates, extended AFDX can be better or worse than DRR-compliant AFDX in terms of delay bounds. But contrary to DRR, the BLS never degrades the RC delay bounds compared to the current AFDX (SP).

4.5 Conclusion

In this chapter, we have detailed the worst-case timing analysis of our extended AFDX incorporating the BLS, using Network Calculus. Then, we have analysed the sensitivity and tightness of the proposed model, in addition to its accuracy compared to the CPA model. Moreover, we have assessed the performance of our solution, compared to an other promising architecture implementing the DRR and the standard one implementing the NP-SP.

The sensitivity analysis has shown that BW has a strong impact on the SCT and RC delay bounds, contrary to L_M and L_R. Concerning the tightness, the different scenarios have demonstrated that the gap between the AWC and analytical NC model is usually below 15%, which highlights its accuracy.

The comparison with the CPA model for SCT confirms that we have solved the optimism issue of the CPA model, and most of the pessimism issues too. Concerning RC traffic, both models have very close delay bounds except for large values of either L_M or L_R.

Finally, we have shown that the extended AFDX offers a better SCT schedulability and often better RC delay bounds than the DRR-compliant AFDX.

However, we note situations where the tightness of the NC model could be improved: when L_M is close to L_R. Additionally, the sensitivity analysis has highlighted the importance of selecting adequate BLS parameters. Therefore, in the next chapter, we will detail an improved NC model to enhance delay bounds tightness and define two BLS parameter tuning methods to enhance the schedulability.

CHAPTER

FIVE

PERFORMANCE ENHANCEMENT

"Even if a scientific model, like a car, has only a few years to run before it is discarded, it serves its purpose for getting from one place to another."

-David L. Wingate

Contents

5.1 Introduction

In the previous chapter, we have highlighted some performance limitations of the Window-based Approach, in terms of delay bound pessimism. More specifically, the evaluation results have shown that the delay bounds are not that tight when L_R is close to L_M. Additionally, through a sensitivity analysis of the window-based model done by varying the different BLS parameters, we have highlighted the impact of BLS parameterisation to enforce the flow constraints.

In this chapter, we aim to overcome these limitations and improve the extended AFDX performances. First, we propose a new BLS modelisation, called Continuous Credit-based Approach (CCbA) to improve the delay bounds tightness. Then, we define a BLS parameter tuning method to improve both RC delay bounds and schedulability. Finally, we conduct a comparative analysis of both analytical models (CCbA vs WbA) in terms of tightness, and a performance evaluation of the extended AFDX when using CCbA and the BLS parameter tuning method, in reference to the DRR and SP as done in the previous chapter.

Results show that the new CCbA model solves the main limitations of the WbA; thus, the SCT and RC delay bounds are tighter with CCbA than with WbA. Concerning the tuning method, the extended AFDX with optimised parameters leads to a larger schedulability enhancement, compared to both SP and DRR.

5.2 Improving the BLS modelisation: the Continuous-Credit-based Approach (CCbA)

In this section, we propose a new BLS model (CCbA) to overcome the limitations of the WbA model. First, we identify the origin of the WbA limitations. Afterwards, we deal with the identified WbA limitations by introducing the improved model, CCbA.

5.2.1 Identification of WbA limitations

The inherent idea of the WbA is based on the different possible combinations of idle and sending BLS windows to model the minimum and maximum service curves. However, when taking a closer look at the credit behaviour of the BLS covering the worst-case scenario of the minimum service curve, we found out the origin of the pessimism inherent to the WbA model. We illustrate this behaviour in Fig.5.1, where the windows introduce a credit discontinuity, which is not a realistic behaviour. Moreover, we have also noticed a similar discontinuity when studying the best-case scenario of the maximum service curve, as shown in Fig.5.2.

We can notice that the discontinuity of the BLS credit happens between the end of the idle window and the start of the sending window for both the minimum and maximum service curves. This issue is situated around L_R. This highlights the fact that L_R is not taken into account in an accurate way by the WbA model, detailed in Chapter 4.

Last but not least, there is an issue when the high priority $p_H(k)$ of a BLS class k is not the highest priority. We denote $\alpha_{H,k}(t) = r_{H,k} \cdot t + b_{H,k}$ the aggregated higher priority traffics of HC(k). In the cases where the BLS high priority $p_H(k)$ is not the highest, the service left by the priorities strictly higher than $p_H(k)$ is $C \cdot t - \alpha_{H,k}(t)$. The resulting rate of the strict minimum service curve β_k^{bls} of the considered BLS class k with the WbA model, denoted $R_{\beta,k}^{bls}$, can be computed using Eq.(4.4) in Theorem 8:

$$R^{bls}_{\beta,k} = \frac{\beta(\Delta^{k,min}_{send})}{\beta(\Delta^{k,min}_{send}) + \beta(\Delta^{k,max}_{idle})} \cdot (C - r_{H,k}) = \frac{(C - r_{H,k}) \cdot \Delta^{k,min}_{send} - b_{H,k}}{(C - r_{H,k}) \cdot \left(\Delta^{k,min}_{send} + \Delta^{k,max}_{idle}\right) - 2 \cdot b_{H,k}} \cdot (C - r_{H,k})$$

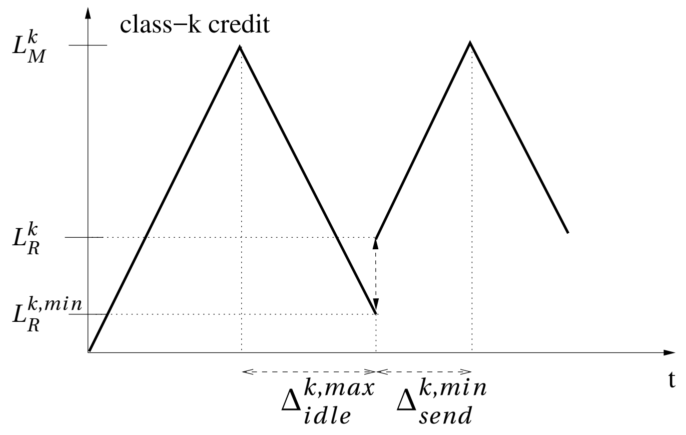


Figure 5.1: WbA: discontinuities with β^{bls}_k windows

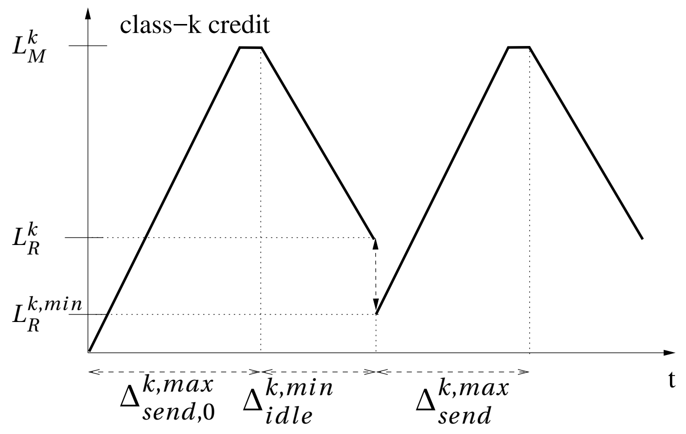


Figure 5.2: WbA: discontinuities with γ^{bls}_k windows

Hence, the burst of the higher priority traffic $b_{H,k}$ has an inherent impact on the minimum service rate. As a consequence, when this burst is too large, the service rate may tend toward 0. This fact may induce very pessimistic delay bounds for class k.

When several BLS are considered, one of them obviously does not have the highest priority (since each BLS class has its own low and high priorities), which shows again the pessimism of such assumption.

Hence, our aim is to handle these identified limitations of the WbA model to better take into account the continuity of the BLS credit and fix the issue of the minimum service rate.

5.2.2 Improving BLS model: the Continuous-Credit-based Approach (CCbA)

We detail here the computation of BLS service curves offered to a BLS class k. The main idea is to compute the consumed and the gained credits. Knowing that the credit is continuous and always between 0 and L_M^k, we use the sum of the consumed and gained credits to compute the minimum and maximum service curves of the BLS node. The main difficulty consists in computing the traffic sent during saturation times, i.e., when the credit is neither gained nor consumed due to the minimum and maximum levels, 0 and L_M^k, respectively.

The strict minimum and maximum service curves offered to a BLS class k by a BLS node are defined in Theorem 8 and in Theorem 9, respectively.

Theorem 8 (Strict Minimum Service Curve offered to a BLS class k by a BLS node)**.** *Consider a server with a constant rate C, implementing BLS shapers. The traffic of class k crosses this server and is shaped by the BLS. Class k has a high priority denoted $p_H(k)$ and a low priority denoted $p_L(k)$ (with $p_L(k) > p_H(k)$). We call $HC(k)$ the traffic classes with a priority strictly higher than $p_H(k)$ and MC(k) the classes with a priority between $p_L(k)$ and $p_H(k)$. The strict minimum service curve guaranteed to the BLS class k is as follows:*

$$\beta_k^{bls}(t) = \left(C - \sum_{h \in HC(k)} r_h - \frac{MFS_k^{sat}}{\Delta_{inter}^{k,\beta}}\right) \cdot \frac{I_{idle}^k}{C} \cdot \left(t - \Delta_{idle}^{k,\beta}\right)^+$$

where

$$\begin{aligned}
MFS_k^{sat} &= \max\left(\max_{j \in MC(k)} MFS_j - \frac{C}{I_{idle}^k} \cdot L_R^k, 0\right) \\
\Delta_{inter}^{k,\beta} &= \frac{L_M^k - L_R^{k,min}}{I_{send}^k} + \frac{L_M^k - L_R^k}{I_{idle}^k} + \frac{\max_{j \in MC(k)} MFS_j}{C} \\
L_R^{k,min} &= \max\left(L_R^k - \frac{\max_{j \in MC(k)} MFS_j}{C} \cdot I_{idle}^k, 0\right) \\
\Delta_{idle}^{k,\beta} &= \frac{L_M^k - L_R^k}{I_{idle}^k} + \frac{\max_{j \in MC(k)} MFS_j}{C}
\end{aligned}$$

Proof. We present here only a sketch of proof, the complete proof is available in Appendix 8.5.2. We search a strict minimum service curve defined by a rate-latency curve, i.e., $\beta_k^{bls}(t) = \rho \cdot (t-\tau)^+$ with rate ρ and latency τ.

First, to compute τ, we consider the maximum latency caused by the BLS.

Secondly, to compute ρ, we consider the fact the credit is a continuous function with values between 0 and L_M^k. Consequently, the sum of the gained and consumed credits is upper bounded by L_M^k, and the credit can saturate at 0 or L_M^k. Thus, the credit consumed during a period δ is not simply the product of the credit increasing rate (denoted I_{send}^k) and the transmission time of the output traffic (denoted $\Delta R^*(\delta)$). It is actually the product of I_{send}^k and the output traffic transmitted while the credit is not saturated. The same is true for the gained credit. Hence, the main issue of the proof is the computation of these saturation times. In particular, we compute the maximum saturation for the MC(k) and HC(k) classes, and the minimum saturation for the class k, as illustrated in Fig.5.3. After this, we use the definition of $\beta(t)$ and the limit of $\frac{\Delta R^*(\delta)}{\delta}$ toward infinity to compute ρ.

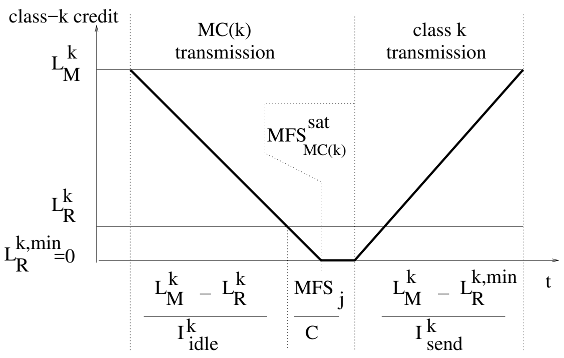


Figure 5.3: Computing $\beta_k^{bls}(t)$

□

Corollary 8 (Strict Minimum Service Curve offered to SCT by a BLS node)**.** *Consider a server with a constant rate C, implementing a BLS shaping the SCT traffic. The SCT, RC and BE traffics cross this server with the following priorities:* $p(SCT) \in \{p_H(SCT) = 0, p_L(SCT) = 2\}$, $p(RC) = 1$, $p(BE) = 3$, *as illustrated in Fig.3.3.*

The strict minimum service curve guaranteed to SCT class is as follows:

$$\beta_{SCT}^{bls}(t) = \left(C - \frac{MFS_{SCT}^{sat}}{\Delta_{inter}^{\beta}}\right) \cdot \frac{I_{idle}}{C} \cdot \left(t - \Delta_{idle}^{\beta}\right)^+$$

where

$$\begin{aligned}
MFS_{SCT}^{sat} &= \max(\max_{j\in RC} MFS_j - \frac{C}{I_{idle}} \cdot L_R, 0)\\
\Delta_{inter}^{\beta} &= \frac{L_M - L_R^{min}}{I_{send}} + \frac{L_M - L_R}{I_{idle}} + \frac{\max_{j\in RC} MFS_j}{C}\\
L_R^{min} &= \max\left(L_R - \frac{\max_{j\in RC} MFS_j}{C} \cdot I_{idle}, 0\right)\\
\Delta_{idle}^{\beta} &= \frac{L_M - L_R}{I_{idle}} + \frac{\max_{j\in RC} MFS_j}{C}
\end{aligned}$$

Proof. We apply Theorem 8 in the particular 3-classes case study presented in Fig.3.1, with SCT as class k.; thus $HC(k) = \emptyset$ and MC(k)=RC. □

Theorem 9 (Maximum Service Curve offered to a BLS class k by a BLS node)**.** *Consider a server with a constant rate C, implementing BLS shapers. The traffic of class k crosses this server and is shaped by the BLS. Class k has a high priority denoted* $p_H(k)$ *and a low priority denoted* $p_L(k)$ *(with* $p_L(k) > p_H(k)$*). We call MC(k) the classes with a priority between* $p_L(k)$ *and* $p_H(k)$*.*

The maximum service curve offered to the class k traffic is as follows. In the absence of backlogged MC(k) traffic: $\gamma_k^{bls}(t) = C \cdot t$*; otherwise, during a backlogged period of MC(k):*

$$\gamma_k^{bls}(t) = \frac{\Delta_{send}^{k,\gamma}}{\Delta_{inter}^{k,\gamma}} \cdot C \cdot t + b_k^{max} \cdot \frac{\Delta_{idle}^{k,\gamma}}{\Delta_{inter}^{k,\gamma}}$$

where

$$b_k^{max} = \frac{C}{I_{send}^k} \cdot L_M^k + MFS_k$$

$$\Delta_{send}^{k,\gamma} = \frac{MFS_k}{C} + \frac{L_M^k - L_R^k}{I_{send}^k}$$

$$\Delta_{idle}^{k,\gamma} = \frac{L_M^k - L_R^k}{I_{idle}^k}$$

and

$$\Delta_{inter}^{k,\gamma} = \Delta_{send}^{k,\gamma} + \Delta_{idle}^{k,\gamma}$$

Proof. We present here only a sketch of proof, the complete proof is available in Appendix 8.5.3. We search a maximum service curve defined by a leaky-bucket curve, i.e., $\gamma_k^{bls}(t) = r \cdot t + b$ with rate r and burst b. First, for r we use the fact that the sum of the gained and consumed credits is lower bounded by $-L_M^k$. Then, we calculate the bounds of saturation times during a period δ. In particular, we calculate the minimum and maximum saturations, as illustrated in Fig.5.4. Finally, we use the definition of $\gamma(t)$ and the limit toward infinity of $\frac{\Delta R^*(\delta)}{\delta}$ to compute r.

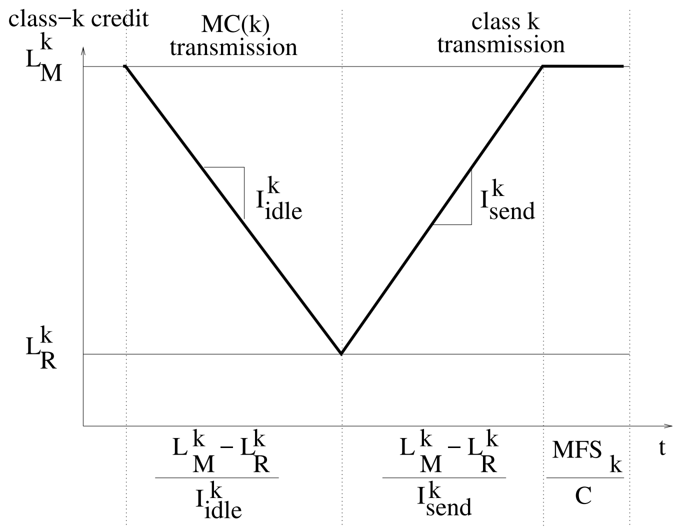


Figure 5.4: Computing $\gamma_k^{bls}(t)$

□

Corollary 9 (Maximum Service Curve offered to SCT by a BLS node)**.** *Consider a server with a constant rate C, implementing a BLS shaping the SCT traffic. The SCT, RC and BE traffics cross this server with the following priorities:* $p(SCT) \in \{p_H(SCT) = 0, p_L(SCT) = 2\}$*,* $p(RC) = 1$*,* $p(BE) = 3$*, as illustrated in Fig.3.3.*

The maximum service curve offered to the SCT traffic class is as follows. In the absence of backlogged RC traffic: $\gamma_{SCT}^{bls}(t) = C \cdot t$*; otherwise, during a backlogged period of RC:*

$$\gamma_{SCT}^{bls}(t) = \frac{\Delta_{send}^{\gamma}}{\Delta_{inter}^{\gamma}} \cdot C \cdot t + b_{SCT}^{max} \cdot \frac{\Delta_{idle}^{\gamma}}{\Delta_{inter}^{\gamma}}$$

where

$$b_{SCT}^{max} = \frac{C}{I_{send}} \cdot L_M + MFS_{SCT}$$

$$\Delta_{send}^{\gamma} = \frac{MFS_{SCT}}{C} + \frac{L_M - L_R}{I_{send}}$$

$$\Delta_{idle}^{\gamma} = \frac{L_M - L_R}{I_{idle}}$$

and

$$\Delta_{inter}^{\gamma} = \Delta_{send}^{\gamma} + \Delta_{idle}^{\gamma}$$

Proof. We apply Theorem 9 in the particular 3-classes case study presented in Fig.3.1, with SCT as class k; thus $HC(k) = \emptyset$ and MC(k)=RC. □

Discussion

First, we consider the fluid traffic assumption, we notice that there is no more credit saturation, neither at 0 nor L_M. For instance, the analysis of the CCbA model in the particular 3-classes case study in Section 3.4.1 shows that: as soon as the credit reaches L_M, the transmission of SCT traffic stops, and as soon as the credit reaches L_R, the transmission of RC traffic stops. As a consequence, $MFS_{SCT}^{sat} = 0$, $MFS_{RC}^{sat} = 0$ and no additional frame when computing the maximum initial latency (for $\beta_{SCT}^{bls}(t)$) or burst (for $\gamma_{SCT}^{bls}(t)$). Based on Appendix 8.4, we can easily show that $\gamma_{SCT}^{bls,fluid}(t) = \gamma_{SCT}^{bls,intui,fluid}(t)$ and $\beta_{SCT}^{bls,fluid}(t) = \beta_{SCT}^{bls,intui,fluid}(t)$. Hence, the CCbA model under the fluid traffic assumption is equivalent to the fluid traffic model of the WbA computed in Section 4.3.2.3.

Secondly, in the particular case where $L_R = 0$, we show in Appendix 8.6 that both models are equal when considering the case study from Section 3.4.1.

Finally, when the BLS queue does not have the highest priority, the minimum service rate under the CCbA model is $\rho = \left(C - \sum_{h \in HC(k)} r_h - \frac{MFS_{MC(k)}^{sat}}{\Delta_{inter}^{\beta}}\right) \cdot \frac{I_{idle}}{C}$. Hence, unlike the WbA (as discussed in Section 5.2.1) the impact of higher priorities on the minimum service rate is only caused by their input rates, not their bursts. This induces a great improvement of the delay bounds under CCbA compared to WbA.

Now that we have discussed the differences between the two models and shown that the CCbA seems very promising, we need to compare both models under varying BLS parameters.

The aim is to quantify the induced enhancement by CCbA in terms of tightness, in reference to the WbA model.

5.2.3 Tightness Analysis of the CCbA model

To compare the two models, we use the same 3-classes case study and scenarios as in Chapter 4. First, concerning $Scenario_{SCT}$ and $Scenario_{RC}$, in both scenarios $L_R = 0$, resulting in identical delay bounds, as detailed in Appendix 8.6. So, we compare WbA and CCbA with the following scenarios:

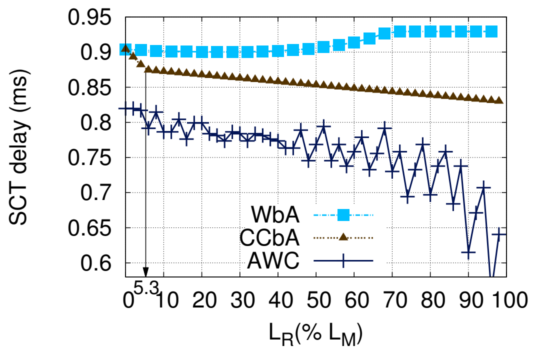
$$Scenario_{LM} = (UR_{SCT} = 20, UR_{RC} = 20, L_M \in [1382.4..216,830], L_R = 1177.6, BW = 0.46)$$

$$Scenario_{LR} = (UR_{SCT} = 20, UR_{RC} = 20, L_M = 22,118, L_R \in [0..0.99] \cdot L_M, BW = 0.46)$$

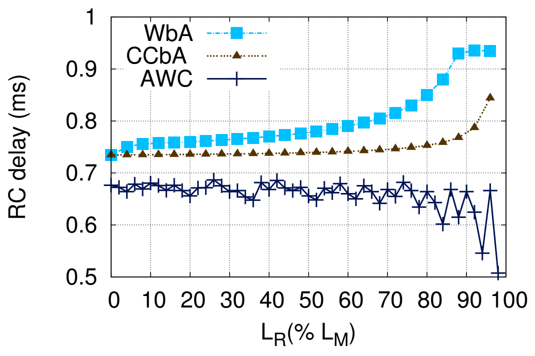
$$Scenario_{BW} = (UR_{SCT} = 20, UR_{RC} = 20, L_M = 22,118, L_R = 1177.6, BW \in [0..0.99])$$

Impact of L_R

SCT and RC delay bounds when varying L_R are shown in Fig.5.5. We notice that CCbA remains firmly below the limit set by $\beta^{sp}_{SCT,2}$, and it is always ruled by $\beta^{bls}_{SCT} \otimes \beta^{sp}_{SCT,0}$. When L_R increases, $L_M - L_R$ decreases, leading to the slow decrease of both CCbA and WbA. Additionally, MFS^{sat}_{RC} decreases until it hits 0 at $L_R = MFS_{RC} \cdot \frac{I_{idle}}{C}$ (according to Theorem 9). This happens in Fig.5.5 at $L_R = 0.053 \cdot L_M$.



(a)



(b)

Figure 5.5: WbA vs CCbA: impact of L_R on: (a) SCT delay bound; (b) RC delay bound with $Scenario_{LR} = (UR_{SCT} = 20, UR_{RC} = 20, L_M = 22,118, L_R \in [0..0.99] \cdot L_M, BW = 0.46)$

Consequently, the tightness of the SCT delay bounds with CCbA is much improved, in reference to the ones with WbA, with the gap remaining around 0.05ms. For instance, around $L_R = 0.8 \cdot L_M$, the CCbA improves the SCT delay bound tightness by 50% compared to WbA.

In Fig5.5, the RC delay bound is ruled by β^{bls}_{RC}. Hence, the RC delay bounds increase when L_R increases with CCbA, but remains below the RC delay bounds of WbA. Consequently, the tightness of the RC delay bound is much improved with CCbA, in reference to the ones with WbA (up to 70% at $L_R = 0.85 \cdot L_M$).

From this analysis, we conclude that the best value for L_R when using CCbA seems to be $L_R = MFS_{RC} \cdot \frac{I_{idle}}{C}$. With no saturation at 0, it gives SCT a good output rate, while limiting the impact on RC traffic. Concerning the differences between WbA and CCbA, we show in Appendix 8.6 that they are identical at $L_R = 0$. This is verified in Fig.5.5.

We can conclude that in reference to WbA, CCbA better takes into account the impact of L_R on the SCT and RC delay bounds, especially when L_R becomes close to L_M. As a result, the delay bounds tightness of SCT (resp.RC) is improved by up to 50% (resp. 70%), compared to WbA.

Impact of L_M

In this second scenario, we have set $L_R = MFS_{RC} \cdot \frac{I_{idle}}{C} = 1177.6$ bits to see the difference between WbA and CCbA (since they are identical for $L_R = 0$).

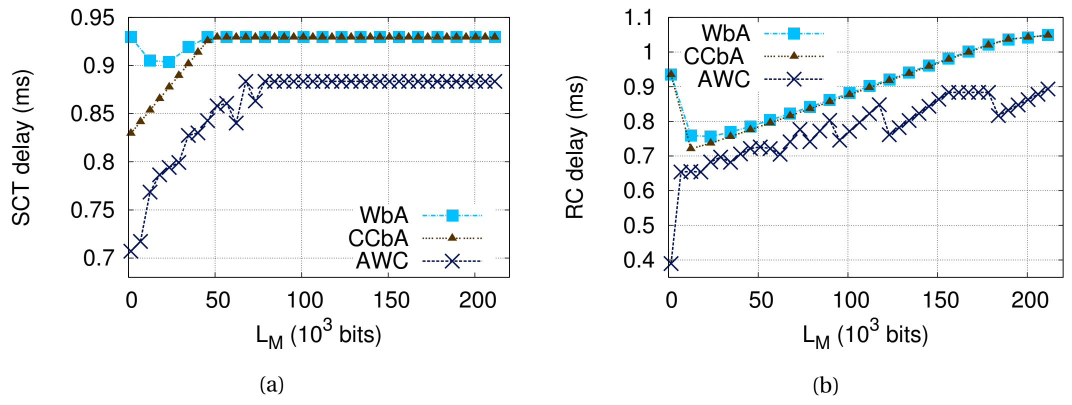


Figure 5.6: WbA vs CCbA: impact of L_M on: (a) SCT delay bound; (b) RC delay bound with $Scenario_{LM} = (UR_{SCT} = 20, UR_{RC} = 20, L_M \in [1382.4..216,830], L_R = 1177.6, BW = 0.46)$

First, concerning the SCT delay bound, we see in Fig.5.6(a) that below $L_M = 50,000$ bits (which correspond to sending 200 consecutive frames between L_R and L_M), the SCT delay bound is ruled by $\beta^{bls}_{SCT} \otimes \beta^{sp}_{SCT,0}$. Hence, with CCbA, when L_M decreases, the rate of the minimum service rate of β^{bls}_{SCT} increases and its initial latency decreases. Consequently, the SCT delay bounds decrease with CCbA when L_M decreases toward L_R. The resulting SCT delay bound tightness is much improved with reference to the WbA model (up to 44%).

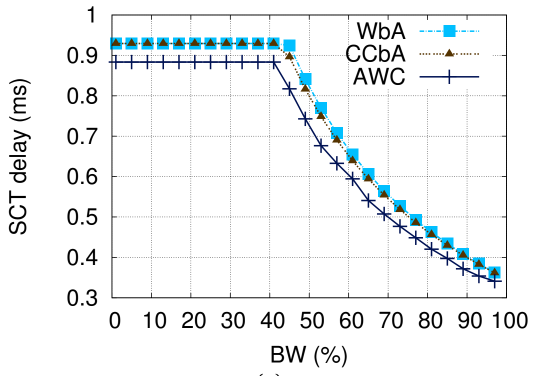
Over $L_M = 50,000$ bits, the SCT delay bound is ruled by $\beta^{sp}_{SCT,2}$, which explains that it is constant and that both models have identical SCT delay bounds.

Concerning RC delay bounds with WbA and CCbA in Fig.5.6(b), they are mostly overlapping, except around $L_M = 7,000$ bits (which corresponds to sending 24 consecutive frames between L_R and L_M) resulting in an improvement of the RC tightness of 50%.

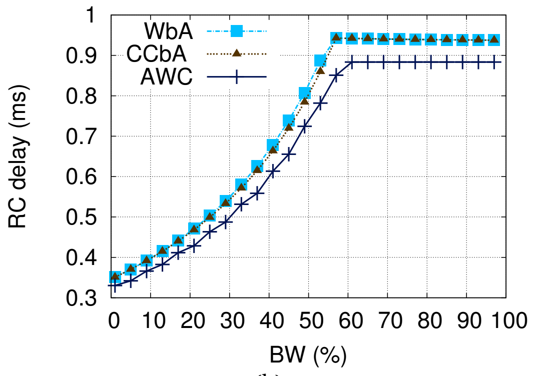
We can conclude that in reference to WbA, the impact of L_M on the SCT delay bounds, when L_M is close to L_R, is better taken into account with CCbA. As a result, the delay bounds tightness of SCT (resp.RC) is improved by up to 44% (resp. 50%), compared to WbA.

Impact of BW

SCT and RC delay bounds when varying BW are shown in Fig.5.7. For both the SCT and RC delay bounds, the WbA and CCbA are overlapping, with the CCbA delay bounds slightly lower than with WbA.



(a)



(b)

Figure 5.7: WbA vs CCbA: impact of BW on: (a) SCT delay bound; (b) RC delay bound with $Scenario_{BW} = (UR_{SCT} = 20, UR_{RC} = 20, L_M = 22,118, L_R = 1177.6, BW \in [0..0.99])$

From this comparative tightness analysis, we can conclude that the tightest model is CCbA, with a maximum distance from AWC usually around 10%, and sometimes up to 20%. In comparison to WbA, the tightness of the SCT (resp. RC) delay bounds is improved up to 50% (resp. 70%).

Now that we have improved the BLS model, we propose in the next part a BLS parameter tuning method to enhance the schedulability.

5.3 BLS parameter tuning

We have noticed that with current AFDX it is easy to guarantee SCT deadlines, but at the expense of increasing the RC delay bounds. On the contrary, with the extended AFDX, we are able to enhance both SCT and RC schedulability and RC delay bounds. Furthermore, we have shown the importance of good BLS parameterisation.

Hence, in this section, we propose a BLS parameter tuning method to improve the schedulability. In the general case, this means parametrising all the BLS parameters in an extended AFDX output port to minimise the impact of higher priorities on lower priorities. Due to the possible interferences between the BLS classes, this seems to be quite difficult in the general case. Additionally, minimising the impact on one class may in turn increase the impact on another class.

So in this section, we study the optimisation problem in the particular 3-classes case described in Fig.3.1 and used in chapters 3, 4 and 5. We propose to tune the BLS parameters to enforce the end-to-end deadlines of the different RC and SCT flows to enhance schedulability, while limiting the impact of the SCT on RC.

5.3.1 Problem formulation

We consider a multi-hop network with individual flows f of class SCT and RC with input arrival curves $\forall\ f, \alpha_f(t) = r_f \cdot t + b_f$.

The aim is to minimise the end-to-end RC flow delay bounds, while fulfilling several constraints. In Section 4.3.1.2, we defined the **rate constraint**, stating the load of a node is lower than minimum guaranteed rate in the considered node. This constraint can be applied to the output port multiplexers and to each traffic class. Hence the constraints are:

1. the **class rate constraint**, stating that in each output port mux, the input rate of an aggregate traffic of class k must be lower than the minimum offered service rate, denoted $R^{mux}_{\beta,k}$;
2. the **aggregate rate constraint**, stating that the load of an output port multiplexer is lower than output link capacity C.
3. the **deadline constraint**, stating that the end-to-end delay $delay^{end2end}_{k,f}$ of a flow f of class k must be lower than its end-to-end deadline $Deadline^{end2end}_{k,f}$.

We call F^{mux}_k the set of flows of a class k in an output port multiplexer mux. The optimisation problem can be formulated as follows: find BLS parameters for each multiplexer mux along the path of each flow $f \in RC$, $\left(L^{mux}_M, L^{mux}_R, BW^{mux}\right)$ such as:

$$\forall \text{ flow } f \in RC\ \forall mux \in path_f \underset{L^{mux}_M, L^{mux}_R, BW^{mux}}{minimize} \left(delay^{end2end}_{RC,f}(L^{mux}_M, L^{mux}_R, BW^{mux})\right)$$

subject to

$\forall$ flow f of class $k \in \{SCT, RC\}$, $\forall mux \in path_f$:

1. $R^{mux}_{\beta,k,f} \geqslant \sum_{f \in F^{mux}_k} r_f$
2. $\sum_{g \in F^{mux}_{SCT}} r_g + \sum_{f \in F^{mux}_{RC}} r_g \leqslant C$
3. $Deadline^{end2end}_{k,f} \geqslant delay^{end2end}_{k,f}(L^{mux}_M, L^{mux}_R, BW^{mux})$

5.3.2 Problem relaxation

Due to the high number of minimisations to be done, one in each output port in each switch, this is a multi-criteria problem and it may not have a solution. Additionally, to solve this problem globally, each set of parameters in each output port needs to be tested for each flow. So, if we consider testing N values for each parameter $(L^{mux}_M, L^{mux}_R, BW^{mux})$, this means having a complexity of $O(l \cdot N^3)$ in an output port; thus $O(l^m \cdot N^{3 \cdot m})$ in the whole network, with m the number of output ports and l the number of flows. This requires of course a tremendous amount of computer power.

In order to drastically reduce this complexity, we will do a per-port analysis for the aggregate traffic of each class. Hence, we will only have to consider one minimisation per port, instead of one minimisation per-port per-RC-flow. Thus, this will reduce the complexity down to $O(m \cdot N^3)$.

Class rate constraint

We keep the input arrival rate of each SCT and RC aggregate traffic, denoted r_{SCT}^{mux} and r_{RC}^{mux}, below the minimum service rate offered to their class in each multiplexer mux, denoted $R_{\beta,SCT}^{mux}$ and $R_{\beta,RC}^{mux}$;

1. $\forall mux \sum_{g \in F_{SCT}^{mux}} r_g = r_{SCT}^{mux} \leqslant R_{\beta,SCT}^{mux}$
2. $\forall mux \sum_{f \in F_{RC}^{mux}} r_f = r_{RC}^{mux} \leqslant R_{\beta,RC}^{mux}$

Deadline constraint

We propose to define a local deadline for class k in the output port multiplexer mux, denoted $Deadline_k^{mux}$, that has to be fulfilled by the aggregate traffic class $k \in \{SCT, RC\}$ in the output port mux:

$$Deadline_k^{mux} \geqslant delay_k^{mux}(L_M^{mux}, L_R^{mux}, BW^{mux}),\ k \in \{SCT, RC\},\ \forall mux,$$

where the $delay_k^{mux}$ is the class-k delay within mux.

The relaxed optimisation problem is then as follows:

$$\forall \text{output port } mux,\ \underset{L_M^{mux}, L_R^{mux}, BW^{mux}}{minimize} \left(delay_{RC}^{mux}(L_M^{mux}, L_R^{mux}, BW^{mux})\right)$$

subject to $\forall mux$:

1. $R_{\beta,k}^{mux} \geqslant r_k^{mux},\ k \in \{SCT, RC\}$
2. $r_{SCT}^{mux} + r_{RC}^{mux} \leqslant C$
3. $Deadline_k^{mux} \geqslant delay_k^{mux}(L_M^{mux}, L_R^{mux}, BW^{mux}),\ k \in \{SCT, RC\}$

Now that we have a problem with an acceptable complexity, we express the objective and the constraints using the parameters of the BLS to assess its linearity.

5.3.3 Constraint propagation

Concerning the class rate constraint, the guaranteed minimum rates of the strict minimum service curves of SCT and RC are defined in Corollary 8 and Corollary 9, respectively. For each service curve β_i^j, we define $R_{\beta,i}^j$ and $T_{\beta,i}^j$ such as $\beta_i^j(t) = R_{\beta,i}^j \cdot (t - T_{\beta,i}^j)^+$. Thus, the class rate constraints are:

$$\begin{aligned} R_{\beta,SCT}^{mux} &= \max(C - r_{RC}^{mux}, \min(C, R_{\beta,SCT}^{bls})) = \max(C - r_{RC}^{mux}, R_{\beta,SCT}^{bls}) \\ R_{\beta,RC}^{mux} &= \max(C - r_{SCT}^{mux}, R_{\beta,RC}^{bls}) \end{aligned}$$

When taking into account the aggregate rate constraint $r_{SCT}^{mux} + r_{RC}^{mux} \leqslant C$, this gives: $r_{SCT}^{mux} \leqslant C - r_{RC}^{mux}$ and $r_{RC}^{mux} \leqslant C - r_{SCT}^{mux}$:

$$\begin{aligned} R_{\beta,SCT}^{mux} &\geqslant \max(r_{SCT}^{mux}, R_{\beta,SCT}^{bls}) \geqslant r_{SCT}^{mux} \\ R_{\beta,RC}^{mux} &\geqslant \max(r_{RC}^{mux}, R_{\beta,RC}^{bls}) \geqslant r_{RC}^{mux} \end{aligned}$$

Hence, the class rate constraint is always fulfilled and can be ignored.

Concerning the deadline constraint, we compute the delay bounds of class $k \in \{SCT, RC\}$ using Theorem 2, with the arrival curve $\alpha_k = \min(C \cdot t, r_k \cdot t + b_k)$ and service curves of sp and bls using Corollaries 6 and 7, respectively. Thus:

$$delay_k^{mux}(L_M^{mux}, L_R^{mux}, BW^{mux}) = \min(delay_k^{bls,mux}, delay_k^{sp,mux})(L_M^{mux}, L_R^{mux}, BW^{mux}) \tag{5.1}$$

with $delay_k^{n,mux}(L_M^{mux}, L_R^{mux}, BW^{mux})$, $n \in \{sp, bls\}$ and $k \in \{SCT, RC\}$ as follows:

$$delay_k^{n,mux}(L_M^{mux}, L_R^{mux}, BW^{mux}) = \frac{b_k + r_k \cdot \frac{b_k}{n_k^{links} \cdot C - r_k}}{R_{\beta,k}^n} + T_{\beta,k}^n - \frac{b_k}{n_k^{links} \cdot C - r_k}$$

So finally we obtain:

$$\forall \text{output port } mux, \underset{L_M^{mux}, L_R^{mux}, BW^{mux}}{minimize} \left(delay_{RC}^{mux}(L_M^{mux}, L_R^{mux}, BW^{mux}) \right)$$

subject to:

1. $r_{SCT}^{mux} + r_{RC}^{mux} \leqslant C$
2. $Deadline_k^{mux} \geqslant delay_k^{mux}(L_M^{mux}, L_R^{mux}, BW^{mux})$, $k \in \{SCT, RC\}$

This is a non-linear problem, with complex functions defining the delays and the rates. There are many ways of solving such a problem numerically: brute force method, random search or heuristics for example. In our case, we will solve this problem based on heuristics taking advantages from the sensitivity analysis done in Section 5.2.3, to select heuristics for L_M^{mux} and L_R^{mux}.

Computing L_R^{mux}:

First, from the sensitivity analysis in Fig.5.5, we have noticed that the most promising value for L_R^{mux} leading to a good SCT output rate while limiting the impact on RC traffic is:

$$L_R^{mux} = MFS_{RC} \cdot BW^{mux} \tag{5.2}$$

Computing L_M^{mux}

Second, we have shown in Fig.5.6, when L_M^{mux} increases, so does the SCT delay bound until it becomes constant thanks to $\beta_{SCT,2}^{sp}$ (see Theorem 6). Moreover, from Fig.5.6(b), we can see that the RC delay bound starts by decreasing, then it increases. So, our aim is to find the expression of L_M^{mux} at this inflection point, which is on the BLS part of the delay bound. Hence, the objective function to minimise is as follows:

$$\begin{aligned} delay_{RC}^{mux}(L_M^{mux}, L_R^{mux}, BW^{mux}) &= \frac{b_{RC} + r_{RC}^{mux} \cdot \frac{b_{RC}}{n_{RC}^{links} \cdot C - r_{RC}^{mux}}}{R_{\beta,RC}^{bls}} + T_{\beta,RC}^{bls} - \frac{b_{RC}}{n_{RC}^{links} \cdot C - r_{RC}^{mux}} \\ &= \frac{b_{RC} + r_{RC}^{mux} \cdot \frac{b_{RC}}{n_{RC}^{links} \cdot C - r_{RC}^{mux}}}{R_{\beta,RC}^{bls}} + \frac{\max_{k \in \{SCT \cup RC \cup BE\}} MFS_k}{R_{\beta,RC}^{bls}} \\ &\quad + \frac{L_M^{mux}}{(1 - BW^{mux}) \cdot C} + \frac{MFS_{SCT}}{C} - \frac{b_{RC}}{n_{RC}^{links} \cdot C - r_{RC}^{mux}} \end{aligned}$$

We denote $A = b_{RC} + r_{RC}^{mux} \cdot \frac{b_{RC}}{n_{RC}^{links} \cdot C - r_{RC}^{mux}} + \max_{k \in \{SCT \cup RC \cup BE\}} MFS_k$; thus:

$$\begin{aligned} delay_{RC}^{bls,mux}(L_M^{mux}, L_R^{mux}, BW^{mux}) &= \frac{A}{(1 - BW^{mux}) \cdot C} \cdot \left(1 + \frac{BW^{mux} \cdot (1 - BW^{mux}) \cdot MFS_{SCT}}{L_M^{mux} - L_R^{mux}}\right) \\ &\quad + \frac{L_M}{(1 - BW^{mux}) \cdot C} + \frac{MFS_{SCT}}{C} - \frac{b_{RC}}{n_{RC}^{links} \cdot C - r_{RC}^{mux}} \end{aligned}$$

To find the inflection point in Fig.5.6, we derive $delay_{RC}^{bls,mux}(L_M^{mux}, L_R^{mux}, BW^{mux})$ to find the null point of the derived function:

$$\begin{aligned} \frac{d(delay_{RC}^{bls,mux}(L_M^{mux}, L_R^{mux}, BW^{mux}))}{d(L_M^{mux})} &= \frac{A}{(1 - BW^{mux}) \cdot C} \cdot \left(-\frac{BW^{mux} \cdot (1 - BW^{mux}) \cdot MFS_{SCT}}{(L_M^{mux} - L_R^{mux})^2}\right) \\ &\quad + \frac{1}{(1 - BW^{mux}) \cdot C} \\ &= -\frac{A \cdot BW^{mux} \cdot MFS_{SCT}}{C \cdot (L_M^{mux} - L_R^{mux})^2} + \frac{1}{(1 - BW^{mux}) \cdot C} = 0 \\ \Rightarrow (L_M^{mux} - L_R^{mux})^2 &= (1 - BW^{mux}) \cdot C \cdot \frac{A \cdot BW^{mux} \cdot MFS_{SCT}}{C} \\ &= A \cdot (1 - BW^{mux}) \cdot BW^{mux} \cdot MFS_{SCT} \end{aligned}$$

$$\Rightarrow L_M^{mux} = MFS_{RC} \cdot BW^{mux} + \sqrt{A \cdot (1 - BW^{mux}) \cdot BW^{mux} \cdot MFS_{SCT}} \qquad (5.3)$$

Therefore, using Eq.(5.2), Eq.(5.3), the relaxed optimisation problem is as follows:

$$\forall \text{output port } mux,\ \underset{BW^{mux}}{minimize}\big(delay_{RC}^{mux}(BW^{mux})\big)$$

$$\begin{aligned} \text{with: } L_R^{mux} &= MFS_{RC} \cdot BW^{mux} \\ L_M^{mux} &= MFS_{RC} \cdot BW^{mux} + \sqrt{A \cdot (1 - BW^{mux}) \cdot BW^{mux} \cdot MFS_{SCT}} \end{aligned}$$

subject to $\forall mux$:

1. $R_{\beta,k}^{mux} \geqslant r_k^{mux}$, $k \in \{SCT, RC\}$
2. $r_{SCT}^{mux} + r_{RC}^{mux} \leqslant C$
3. $Deadline_k^{mux} \geqslant delay_k^{mux}(BW^{mux})$, $k \in \{SCT, RC\}$

5.3.4 Solving the problem

To compute BW^{mux}, we propose Algorithm 2, which takes into account as inputs $Deadline_{SCT}^{mux}$ and $Deadline_{RC}^{mux}$. We use a loop to compute the possible values for BW^{mux} in Line 2. Inside the loop, we compute the corresponding values of L_R^{mux}, i.e., lr, and L_M^{mux}, i.e., lm, in Lines 4 and 5. Then with Eq.(5.1), we compute the SCT and RC delay bounds in Lines 6 and 7. Next, in Line 8, we verify the local deadlines conditions. If they are fulfilled, we store the delays and bw in Outputs, in Line 9. Finally, after testing all the bw in the loop, we select the BW^{mux} leading the the minimum RC delay bounds, in Line 13. If no BW^{mux} fulfils the condition, we return $+\infty$ for each delay bound.

As we can notice, we need to define both local deadlines of SCT and RC within mux to enable Algorithm2. Hence, we have defined two methods to compute these local deadlines: Heuristic Deadline (HD) and Dichotomous Deadline (DD) methods.

Heuristic Deadline Method

We propose to set the local deadline of class k in the output port multiplexer mux as the product of the sum of the multiplexer deadlines, denoted $\sum deadline_k^{mux}$, and the weight of class k rate going through multiplexer mux, relatively to the global class k rate in all the multiplexers m along the path $path_f$, thus:

$$Deadline_k^{mux} = \frac{\sum\limits_{flw \in F_k^{mux}} r_{flw}}{\sum\limits_{m \in path_f} \sum\limits_{flw \in F_k^m} r_{flw}} \cdot \sum deadline_k^{mux}$$

Using Eq.(4.1) in Section 4.3.1.2 we have:

$$\sum_{sw \in path_f} delay_{k,f}^{sw} \leqslant Deadline_{k,f}^{end2end} - delay_{k,f}^{es} - delay_{k,f}^{prop}$$

with $delay^{sw}_{k,f} = \frac{MFS_f}{C} + 1\mu s + delay^{mux}_{k,f}$

$$\begin{aligned}\Rightarrow \sum_{mux \in path_f} delay^{mux}_{k,f} \leqslant &\quad Deadline^{end2end}_{k,f} - \sum_{mux \in path_f}\left(\frac{MFS_f}{C} - 1\mu s\right) \\ &\quad -delay^{es}_{k,f} - delay^{prop}_{k,f} \\ \leqslant &\quad \sum deadline^{mux}_{k}\end{aligned}$$

$$\Rightarrow \sum deadline^{mux}_{k} = \min_{m \in path_f, f \in F^{mux}_k} \left\{ \begin{aligned} & Deadline^{end2end}_{k,f} - \sum_{m \in path_f}\left(\frac{MFS_f}{C} - 1\mu s\right) \\ & -delay^{es}_{k,f} - delay^{prop}_{k,f} \end{aligned}\right\}$$

We use the minimum to reduce the complexity, even though this strengthens the constraint. This method has the benefice of being simple to use. However, it has the potential flaw of imposing a local deadline to class k delay bound in a multiplexer, without checking that the delay bounds in the other switches are able or not to reach their own deadlines. To cope with these limitations, we propose the second computation method, DD method described, in Algorithm 3.

Algorithm 2 BLS parameters algorithm in a multiplexer mux knowing the local deadlines:Delays&BLSparams()

```
Input: $Deadline^{mux}_{SCT}$; $Deadline^{mux}_{RC}$ $b_{SCT}$;$r^{mux}_{SCT}$;$MFS_{SCT}$;$b_{RC}$;$r^{mux}_{RC}$;$MFS_{RC}$;$MFS_{BE}$;
Output: $BW^{mux}$,$delay^{mux}_{SCT}$,$delay^{mux}_{RC}$
 1: Data=[$Deadline^{mux}_{SCT}$; $Deadline^{mux}_{RC}$; $b_{SCT}$;$r^{mux}_{SCT}$;$MFS_{SCT}$;$b_{RC}$;$r^{mux}_{RC}$;$MFS_{RC}$;$MFS_{BE}$]
 2: for bw ∈ [0.001 : 0.001 : 0.999] do
 3:     $K = \sqrt{(b_{SC} + r_{ST} \cdot \frac{b_{RC}}{n^{links}_{RC} \cdot C - r^{mux}_{RC}} + \max_{k \in \{SCT \cup RC \cup BE\}} MFS_k) \cdot (1 - bw) \cdot bw \cdot MFS_{SCT}}$
 4:     $lr = MFS_{RC} \cdot bw$
 5:     $l_m = l_r + K$
 6:     $d^{mux}_{SCT}$= $Delay^{mux}_{SCT}$(Data, $bw$, $l_m$, $l_r$)
 7:     $d^{mux}_{RC}$= $Delay^{mux}_{RC}$(Data, $bw$, $l_m$, $l_r$)
 8:     if $d^{mux}_{SCT} \leqslant Deadline^{mux}_{SCT}$ and $d^{mux}_{RC} \leqslant Deadline^{mux}_{RC}$ then
 9:         Outputs.add($bw$,$d^{mux}_{RC}$,$d^{mux}_{SCT}$)
10:     end if
11: end for
12: if notEmpty(Outputs) then
13:     [$BW^{mux}$,$delay^{mux}_{RC}$,$delay^{mux}_{SCT}$]=Outputs.get(IndexOfMinDRCs(Outputs))
14: else
15:     [$BW^{mux}$,$delay^{mux}_{SCT}$,$delay^{mux}_{SCT}$]=[-,+∞,+∞] %no admissible parameters
16: end if
```

Algorithm 3 Local deadline computation algorithm with dichotomous method, along the path of flow f: DichotomousDeadline()

```
Input: $\forall mux \in path_f, \forall g \in F_k^{mux}, k \in \{SCT, RC\}, MFS_g, r_g, b_g, \sum deadline_k^{mux}$
Output: $\forall mux, Deadline_{SCT}^{mux}, Deadline_{RC}^{mux}$
 1: Data=[$\forall mux \in path_f$, $Deadline_{RC}^{mux}$; $\forall mux \in path_f, \forall g \in F_k^{mux}, k \in \{SCT, RC\}$,
    $MFS_g, r_g, b_g$]
 2: for $\forall mux$ in $path_f$ do
 3:   $Deadline_{RC}^{mux}$= RCHeuristicDeadline($\sum deadline_{RC}^{mux}$,data)
 4:   $Deadline_{SCT}^{under,mux}$= SCTHeuristicDeadline($\sum deadline_{SCT}^{mux}$,data);
 5:   $Deadline_{SCT}^{over,mux} = \sum deadline_{SCT}^{mux}$
 6:   $delay_{SCT,}^{under,mux}$=Delays&BLSparams($Deadline_{SCT}^{under,mux}$,$data$).delay
 7:   $delay_{SCT}^{over,mux}$=Delays&BLSparams($Deadline_{SCT}^{ove,nmux}$,$data$).delay
 8: end for
 9: $\sum deadline_{SCT}^{over,mux} = \sum_{mux \in path_f} Deadline_{SCT}^{over,mux}$
10: $\sum deadline_{SCT}^{under,mux} = \sum_{mux \in path_f} Deadline_{SCT}^{under,mux}$
11: if $\sum_{mux \in path_f} delay_{SCT}^{over,mux} \leqslant \sum deadline_{SCT}^{mux}$ then
12:   return $\forall mux \in path_f, Deadline_{RC}^{mux}, Deadline_{SCT}^{over,mux}$
13: else if $\sum_{\forall mux \in path_f} delay_{SCT}^{over,mux} \geqslant \sum deadline_{SCT}^{mux} \geqslant \sum_{\forall mux \in path_f} delay_{SCT}^{under,mux}$
    then
14:   while $\sum deadline_{SCT}^{mux} - \sum_{mux \in path_f} delay_{SCT}^{under,mux} \geqslant \epsilon$ do
15:     $\sum deadline_{SCT}^{cur,mux} = (\sum deadline_{SCT}^{over,mux} + \sum deadline_{SCT}^{under,mux})/2$
16:     for $\forall mux$ in $path_f$ do
17:       $Deadline_{SCT}^{cur,mux}$= SCTHeuristicDeadline($\sum deadline_{SCT}^{cur,mux}$, data);
18:       $delay_{SCT}^{cur,mux}$=Delays&BLSparams($Deadline_{SCT}^{cur,mux}$,$data$).delay
19:     end for
20:     if $\sum_{\forall mux \in path_f} delay_{SCT}^{cur,mux} \geqslant \sum deadline_{SCT}^{mux}$ then
21:       $\sum deadline_{SCT}^{over,mux} = \sum deadline_{SCT}^{cur,mux}$
22:     else
23:       $\sum deadline_{SCT}^{under,mux} = \sum deadline_{SCT}^{cur,mux}$
24:       $\forall mux \in path_f, Deadline_{SCT}^{under,mux} = Deadline_{SCT}^{cur,mux}$
25:     end if
26:   end while
27: end if
28: return $\forall mux \in path_f, Deadline_{RC}^{mux}, Deadline_{SCT}^{under,mux}$
```

Dichotomous Deadline Method

The main idea of this second method is to compute a less constrained local deadline for SCT than the one with HD. This fact will give more improvement margins for RC delay bounds. Hence, to simplify the computation, we use the heuristic local deadline for RC and we propose to use the dichotomous method (described in Algorithm 3) for SCT. To compute the delay bound of class SCT along the path of a flow f, we will use two deadline values: one leading to delay bounds equal or lower to the deadline, and one leading to delay bounds equal or higher than the deadline.

The first deadline is computed with the heuristic method, and may give a SCT lower bound.

$$Deadline_{SCT}^{under,mux} = \frac{\sum\limits_{flw \in F_{SCT}^{mux}} r_{flw}}{\sum\limits_{m \in path_f} \sum\limits_{flw \in F_{SCT}^{m}} r_{flw}} \cdot \sum deadline_{SCT}^{mux}$$

To obtain a higher SCT delay bound, we consider the following SCT deadline:

$$Deadline_{SCT}^{over,mux} = \sum deadline_{SCT}^{mux}$$

The Dichotomous Deadline (DD) method is detailed in Algorithm 3. The objective is to find $\sum deadline_{SCT}^{cur,mux}$ such as:

$$\sum_{mux \in path_f} delays(Deadline_{SCT}^{cur,mux}) = \sum deadline_{SCT}^{mux}$$

with $Deadline_{SCT}^{cur,mux}$ = SCTHeuristicDeadline($\sum deadline_{SCT}^{cur,mux}$), as illustrated in Fig.5.8.

From Line 2 to Line 10, we initialise the dichotomous search. In Lines 3, 4 and 5, we compute the initial Deadlines, i.e., $Deadline_{RC}^{mux}$, $Deadline_{SCT}^{over,mux}$ and $Deadline_{SCT}^{under,mux}$ for each mux. This leads to Lines 6 and 7, to the computation of the corresponding SCT delay bounds $delay_{SCT}^{over,mux}$ and $delay_{SCT}^{under,mux}$ using Algorithm 2. After all the mux have been considered, we compute in Lines 9 and 10 the two dichotomous variables: $\sum deadline_{SCT}^{over,mux}$ and $\sum deadline_{SCT}^{over,mux}$, the sum of the deadlines leading to SCT delay bounds under or over the sum of the multiplexer deadlines $\sum deadline_{SCT}^{mux}$, as illustrated in Fig.5.8.

Then in Line 11, we check whether the value $\sum_{mux \in path_f} delay_{SCT}^{over,mux}$ is actually over $\sum deadline_{SCT}^{mux}$. If not, we return $Deadline_{SCT}^{over,mux}$ since a dichotomous search is not possible. Else, we start the dichotomous search as illustrated in Fig.5.8, where $\sum deadline_{SCT}^{mux}$ is bounded by $\sum_{mux \in path_f} delay_{SCT}^{over,mux}$ and $\sum_{mux \in path_f} delay_{SCT}^{under,mux}$.

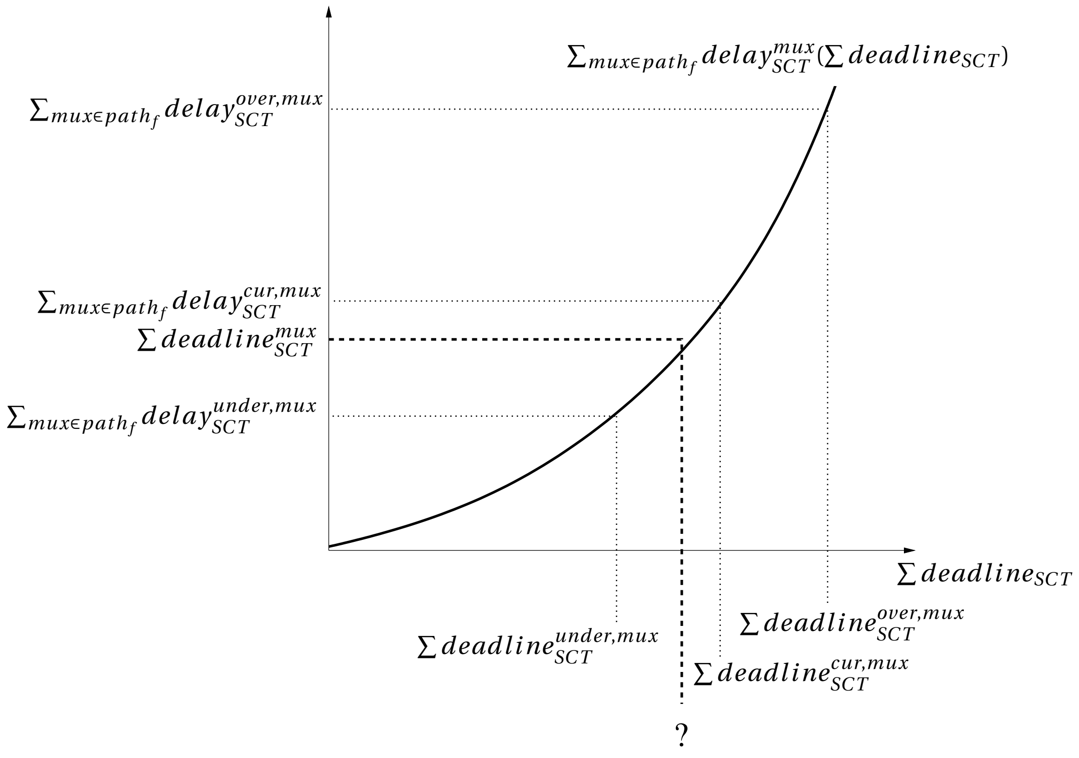
In Line 14, we set the stop condition using an $\epsilon << 0$ such as:

$$\sum deadline_{SCT}^{mux} - \sum_{mux \in path_f} delay_{SCT}^{under,mux} \geqslant \epsilon$$

The objective is to find a solution as close as possible to the $\sum deadline_{SCT}^{mux}$ and respecting the deadline constraint.

Then, we start the next iteration in Line 15, by computing the current $\sum deadline_{SCT}^{cur,mux}$. We use it to compute the local deadlines in each mux and the resulting SCT delay bounds in Lines 17 and 18.

The final steps consist in determining whether $\sum_{mux \in path_f} delay_{SCT}^{cur,mux}$ (the sum of the current delay bounds), is under or over $\sum deadline_{SCT}^{mux}$ in Line 20. Then, we redefine the values of the current loop, either $\sum deadline_{SCT}^{over,mux}$ in Line 21, or $\sum deadline_{SCT}^{under,mux}$ and $Deadline_{SCT}^{under,mux}$ in Lines 23 and 24.



$\sum_{mux \in path_f} delay_{SCT}^{mux}(\sum deadline_{SCT})$
$\sum_{mux \in path_f} delay_{SCT}^{over,mux}$
$\sum_{mux \in path_f} delay_{SCT}^{cur,mux}$
$\sum deadline_{SCT}^{mux}$
$\sum_{mux \in path_f} delay_{SCT}^{under,mux}$
$\sum deadline_{SCT}$
$\sum deadline_{SCT}^{under,mux}$
$\sum deadline_{SCT}^{over,mux}$
$\sum deadline_{SCT}^{cur,mux}$
?

Figure 5.8: Dichotomous search of optimum $\sum deadline_{SCT}$

Finally, we obtain the Algorithm 4. First, we compute the local deadlines with either the Heuristic method or the Dichotomous one described in Algorithm 3. Then, we use the function Delays&BLSparams() described in Algorithm 2 to compute the BLS parameters and delay bounds of SCT and RC.

Algorithm 4 BLS parametrisation along the path of flow f

```
Input: local deadline;$\forall mux \in path_f, \forall g \in F_k^{mux}, k \in \{SCT, RC\}, MFS_g, r_g, b_g, \sum deadline_k^{mux}$
Output: $\forall mux, L_M^{mux}, L_R^{mux}, BW^{mux}$
1: if local deadline = heuristic deadline then
2:     $\forall mux$ in $path_f$, $Deadline_{SCT}^{mux}$,$Deadline_{RC}^{mux}$=HeuristicDeadline(Input)
3: else
4:     $\forall mux$ in $path_f$, $Deadline_{SCT}^{mux}$,$Deadline_{RC}^{mux}$=DichotomousDeadline(Input)
5: end if
6: return $\forall mux \in path_f$ =Delays&BLSparams($Deadline_{SCT}^{mux}, Deadline_{RC}^{mux}$,Input)
```

5.3.5 Comparing intuitive parameters to optimised parameters

We only test herein the parameter tuning method on a single-hop network, where:

$$\begin{aligned} Deadline_k^{under,mux} &= \frac{\sum\limits_{flw \in F_k^{mux}} r_{flw}}{\sum\limits_{m \in path_f} \sum\limits_{flw \in F_k^m} r_{flw}} \cdot \sum deadline_k^{mux} \\ &= \sum deadline_k^{mux} = Deadline_k^{over,mux} \end{aligned}$$

Hence, in this single-hop case, heuristic deadline and dichotomous deadline methods lead to the same local deadlines. The full testing on a multi-hop network is done in the validation chapter, Chapter 6.

To show the gain of optimising the BLS parameters, we consider an **intuitive tuning** with L_R such as the saturation at 0 is minimum, and such as 80 frames can be sent consecutively:

$$L_R = MFS_{RC} \cdot BW \text{ and } L_M = 80 \cdot (1 - BW) \cdot MFS_{SCT}.$$

Also, we choose BW such as the reserved bandwidth is equal to the minimally needed bandwidth:

$$BW = UR_{SCT}$$

The SCT and RC delay bounds under $Scenario_{SCT}$ and $Scenario_{SCT}$ are illustrated in Fig.5.9 and Fig.5.10, respectively.

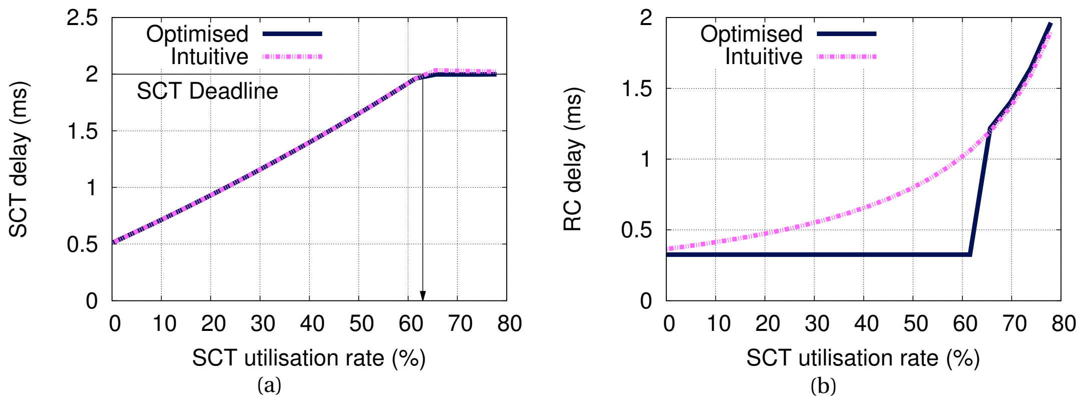


Figure 5.9: Comparing intuitive and optimised BLS when varying SCT maximum utilisation rate with $Scenario_{SCT} = (UR_{SCT} \in [0.1:78], UR_{RC} = 20)$

First, concerning the schedulability, we can see in both Fig.5.9(a) and Fig.5.10(a) that the maximum SCT and RC utilisation rates are much improved with the optimised parameters. For instance, it is almost doubled in Fig.5.10(a), with UR_{RC}^{mux} going from 40% to 79% and up to 24% in Fig.5.9(a), with UR_{SCT} increasing from 62% to 77%.

Secondly, concerning RC delay bounds, both Fig.5.9(b) and Fig.5.10(b) show that the optimised tuning can largely reduce the RC delay bounds, while keeping the SCT schedulability.

For instance, in Fig.5.9 it divides the RC delay bound by 3 at $UR_{SCT} = 60\%$ in comparison to intuitive tuning.

Hence, we have shown that the optimised parameters enable us to obtain an enhanced schedulability and RC delay bounds, without having to go through a time-expensive process like brute force, to find appropriate parameters.

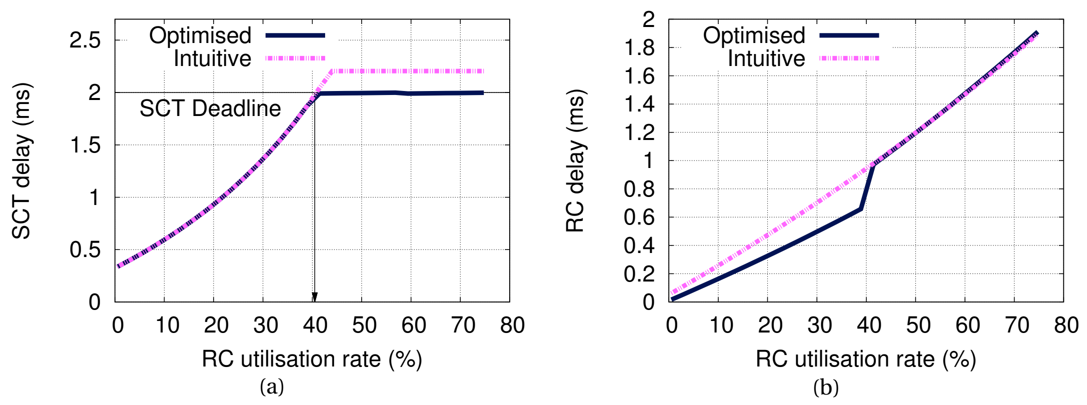


Figure 5.10: Comparing intuitive and optimised BLS when varying RC maximum utilisation rate with $Scenario_{RC} = (UR_{SCT} = 20, UR_{RC} \in [0.5:72])$

5.4 Performance analysis

In this section, we compare the performances of the extended AFDX based on the WbA model in chapter4 (with $L_M = 22,118, L_R = 0, BW = 0.46$) to the performances of the extended AFDX based on the CCbA model in Chapter 5 and enhanced by the HD parameter tuning method, in reference to the current AFDX. Then, we compare the extended AFDX with CCbA and HD tuning method to the DRR-compliant AFDX (also enhanced by the tuning method), in reference to the current AFDX.

We consider the single-hop network defined in Section 3.4.1 with $Scenario_{SCT} = (UR_{SCT} \in [0.1:78], UR_{RC} = 20)$ and $Scenario_{RC} = (UR_{SCT} = 20, UR_{RC} \in [0.5:72])$. Thus, both local deadline computation methods lead to the same results, and WbA and CCbA are identical for $L_R = 0$ as shown in Appendix 8.6.

The aim of the optimisation is to minimise the RC delay bounds. To be able to compare extended (BLS) and DRR-compliant AFDX, we must find a set of DRR parameters that offers similar guarantees. To do so, we change the service curves offered by the BLS in the optimisation problem in Section5.3 by the service curves offered by the DRR implementation (detailed in Section 4.4.4.1). Finally, to solve such an optimisation problem, we use the brute force method with two loops: one for Q_{SCT} and one for Q_{RC}. The SCT and RC delay bounds are shown in Fig.5.11 and Fig.5.12 under the considered scenarios.

Extended AFDX: WbA vs CCbA + tuned parameters

In Fig.5.11(a), results shows that the SCT delay bounds computed with the extended AFDX from chapter 4 do not fulfil the SCT deadline after $UR_{SCT} = 63\%$. Hence, the enhancements of chapter 5 (CCbA and parameter tuning) have improved the SCT schedulability by 25%, from 63% to 79%. Similarly, Fig.5.12(b), we can see that the RC deadline is not fulfilled after $UR_{RC} = 63\%$ under the WbA model defined in Theorem 7, leading to an RC schedulability enhancement of 14% under the CCbA and tuning method, compared to the RC schedulability under WbA.

In Fig.5.11(b) and Fig.5.12(b), the RC delay bounds have also been largely improved by the CCbA model and parameter tuning compared to the ones under the WbA model, while guaranteeing the SCT schedulability. For instance, in Fig.5.11(b), at $UR_{SCT} = 60\%$ the RC delay bound is decreased by 50% under CCbA and the tuning method, compared to the one under WbA. Similarly, at $UR_{RC} = 39\%$ in Fig.5.12(b), the RC delay bound is also decreased by 50% under CCbA and the tuning method, compared to the one under the WbA.

Extended AFDX chap5 vs DRR-compliant

Concerning the SCT and RC schedulability, the extended AFDX of Chapter 5 (CCbA with the parameter tuning) and DRR-compliant AFDX have the same performances.

The difference of performance between extended and DRR-compliant AFDX concerns the RC delay bounds. In Fig.5.11(b), extended AFDX of Chapter 5 and DRR-compliant AFDX are both largely better than the current AFDX. For instance, at $UR_{SCT} = 60\%$, the RC delay bound with current AFDX is 10-times the one under extended AFDX in Chapter 5. Additionally, extended AFDX of chapter 5 is either equal or better than DRR-compliant. In fact, at $UR_{SCT} = 60\%$, the RC delay bound under the DRR-compliant AFDX is three times the delay bound under the extended AFDX with CCbA and tuned parameters in Chapter 5. Interestingly, the extended AFDX is consistently better than the DRR-compliant AFDX, which was not the case in the results of Chapter 4.

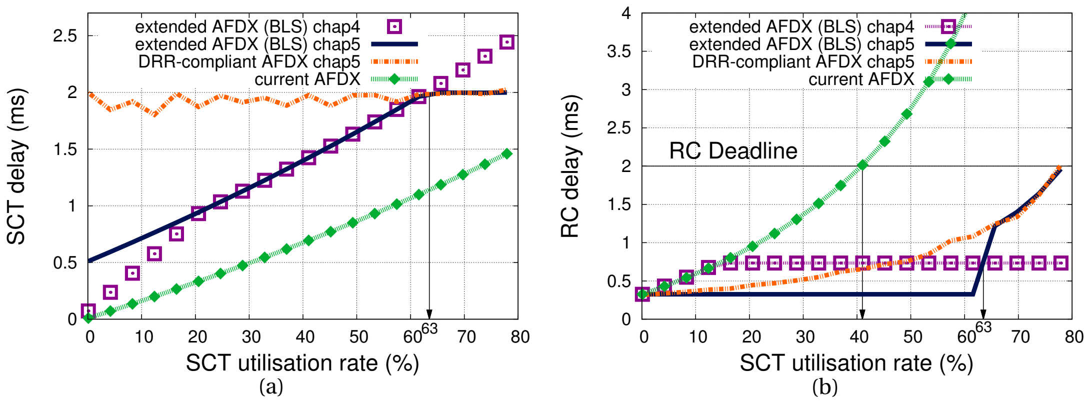


Figure 5.11: Comparing optimised BLS, SP and DRR when varying SCT maximum utilisation rate with $Scenario_{SCT} = (UR_{SCT} \in [0.1 : 78], UR_{RC} = 20)$

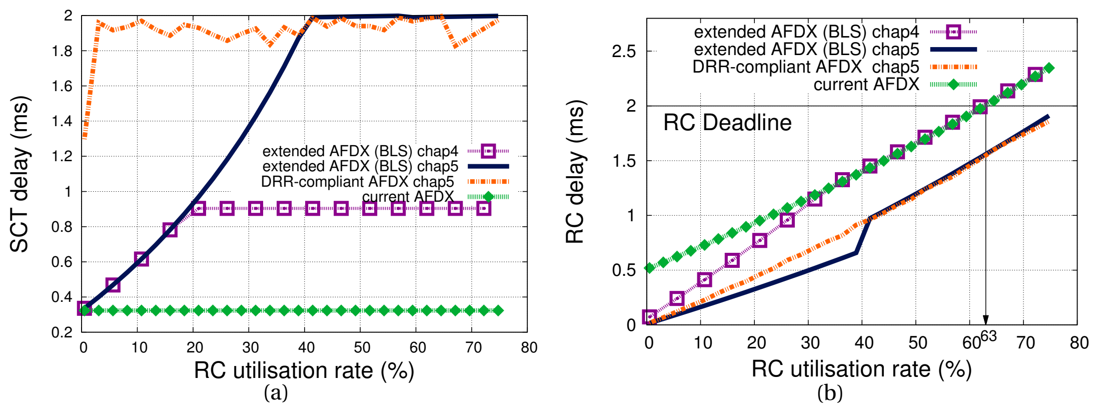


Figure 5.12: Comparing optimised BLS, SP and DRR when varying RC maximum utilisation rate with $Scenario_{RC} = (UR_{SCT} = 20, UR_{RC} \in [0.5 : 72])$

Finally, we conclude that the extended AFDX with CCbA and tuned parameters leads to a large improvement compared to Chapter 4 WbA extended AFDX, in terms of schedulability (up to 27%) and RC delay bounds (up to 50%).

5.5 Conclusion

In this chapter, we have presented a new model for the BLS node and its associated tuning parameters. The results have shown the improvements inherent to the new model (CCbA) over the previous one (WbA) in terms of tightness. In particular, the impact of the BLS parameter L_R and the higher priority bursts are much better taken into account. As a result, the main limitation of the previous model (low tightness when L_R is close to L_M) has been handled.

With the new model, for the 3-classes case study, we have been able to propose a method to compute BLS parameters to obtain a better SCT (resp. RC) schedulability compared to intuitive parameters, up to 49% (resp. 24%). The BLS parameter tuning has also largely enhanced the RC delay bound by dividing it up to three times, compared to intuitive parameters.

We have also shown that the CCbA with tuned BLS parameters improves SCT (resp. RC) schedulability up to 25% (resp. 14%) and RC delay bounds up to 50%, with reference the WbA from chapter 4.

Finally, compared to DRR-compliant AFDX with tuned parameters, the extended AFDX with CCbA and tuned parameters consistently improves the RC delay bounds, up to 66 %.

Now that we have a good BLS modelisation and an appropriate BLS tuning method, in the next chapter we will validate our proposal using realistic avionics case studies.

CHAPTER

SIX

VALIDATION

"There is nothing like looking, if you want to find something."

-J.R.R. Tolkien

Contents

6.1 Introduction

In the previous chapters, we have proposed an extended AFDX to add mixed-criticality to the AFDX. We have also proposed two models, which have been compared to select the tightest one. Then, we have proposed two tuning methods to find the best BLS parameters. Until now, all the verifications have been done using preliminary evaluations on a single-hop network.

In this chapter, we will extend these preliminary evaluations through a generic avionics case study to validate the proposal. In particular, we compare the two deadline computation methods. This had not been possible on the single-hop network since both methods result in the same deadline in this case. Then, we compare DRR-compliant, current and extended

AFDX with both deadline computation methods. Finally, we detail an avionics application, adding A350 flight control traffic to the AFDX.

6.2 Generic Avionics Case study

In this section, we consider a representative and generic avionics network with a multi-hop architecture. The aim is to generalise the conclusions of the previous chapters on the single-hop network to a multi-hop network. We start by presenting the case study. Then, we conduct a comparative analysis of both proposed BLS models, and intuitive vs optimised BLS parameters under both local deadline computation methods. We finish by comparing the delay bounds with the extended, current and DRR-compliant AFDX.

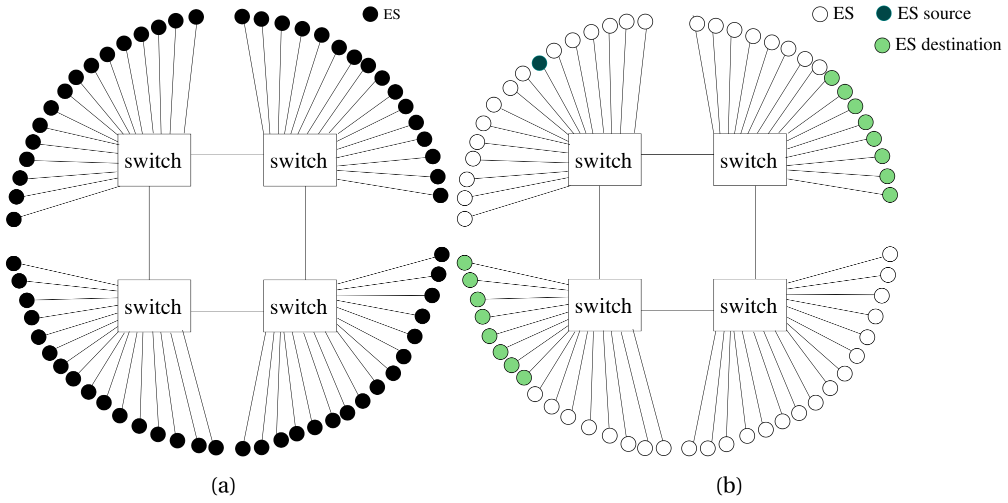


Figure 6.1: Representative AFDX network: (a) Architecture; (b) Traffic communication patterns

6.2.1 Case study

Our case study is a representative avionics communication architecture of the A380, based on a 1-Gigabit AFDX backbone network, which consists of 4 switches and 64 end-systems as shown in Fig. 6.1 (a). Each circulating traffic flow on the backbone network is a multicast flow with 16 destinations, and crosses two successive switches before reaching its final destinations. The first switch in the path receives traffic from 16 end-systems to forward it in a multicast way to its two neighbouring switches. Afterwards, the second switch in the path, which receives traffic from the two predecessor switches, forwards the traffic in its turn to the final end-systems. Each end-system receives data from 16 end-systems. Figure 6.1 (b) shows the traffic communication patterns between the source and the final destinations of a given flow.

In this multi-hop network, each end-system es generates n_i^{es} flows of traffic type $i \in \{SCT, RC, BE\}$. We consider that all end-systems are identical and each generates the same number of flows n_i^{es}.

As a consequence, the utilisation rate in both the first and second switches is the bottleneck utilisation rate for each type of traffic $i \in \{SCT, RC, BE\}$, $UR_i^{bn} = 16 \cdot n_i^{es} \cdot \frac{MFS_i}{BAG_i} \cdot \frac{1}{C}$.

We consider the traffics SCT, RC and BE defined in Table 6.1, and various scenarios similar to the ones defined in the previous chapters and described in Table 6.2.

Priority	Traffic type	MFS	BAG	deadline	jitter
		(Bytes)	(ms)	(ms)	(ms)
0/2	SCT	64	2	2	0
1	RC	320	2	2	0
3	BE	1024	8	none	0.5

Table 6.1: Avionics flow Characteristics

scenario	UR_{SCT}^{bn}(%)	n_{SCT}^{es}	UR_{RC}^{bn}(%)	n_{RC}^{es}	L_M	L_R	BW
	(%)		(%)		(bits)	(bits)	(%)
$scenario_{SCT}$	[0.40...78.6]	[1:3:192]	20.48	10	22,118	0	46
$scenario_{RC}$	20.07	49	[2.05...79.9]	[1:1:39]	22,118	0	46
$scenario_{L_M}$	20.07	49	20.48	10	[1382.4..216,830]	1177.6	46
$scenario_{L_R}$	20.07	49	20.48	10	22,118	[0..0.99]$\cdot L_M$	46
$scenario_{BW}$	20.07	49	20.48	10	22,118	1177.6	[1...99]

Table 6.2: Evaluation scenarios: input traffics and BLS parameters

6.2.2 Comparing Window-based and Continuous Credit-based Approaches

The first validation concerns the comparison of the Window-based (WbA) and Continuous Credit-based Approaches (CCbA). The delay bounds of SCT and RC under the different scenarios, illustrated in Figures 6.2, 6.3, 6.4, 6.5 and 6.6, lead the same conclusion: CCbA outperforms WbA.

The differences between the two models are particularly visible when LR is close to L_M. In Figure 6.4(a), when L_M decreases toward 0, the SCT delay bounds under the WbA increases until 2.3ms; whereas under CCbA, it decreases linearly toward 1.9ms, as shown in Figure 6.5(a). Furthermore, we can observe the same behaviour in Figure 6.5(b), where the RC delay bounds under CCbA remain strictly below the ones under WbA.

Hence, these results confirm that the CCbA model still outperforms the WbA even in a multi-hop network.

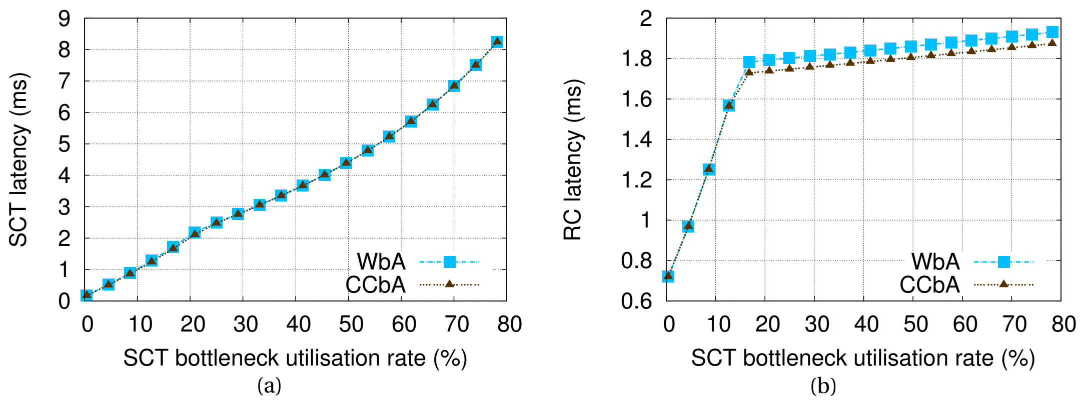


Figure 6.2: Comparing WbA and CCbA, impact of UR^{bn}_{SCT} on: (a) SCT delay bound; (b) RC delay bound, with $Scenario_{SCT} = \left(UR^{bn}_{SCT} \in [0.4:79], UR^{bn}_{RC} = 20.5, L_M = 22,118, L_R = 0, BW = 0.46\right)$

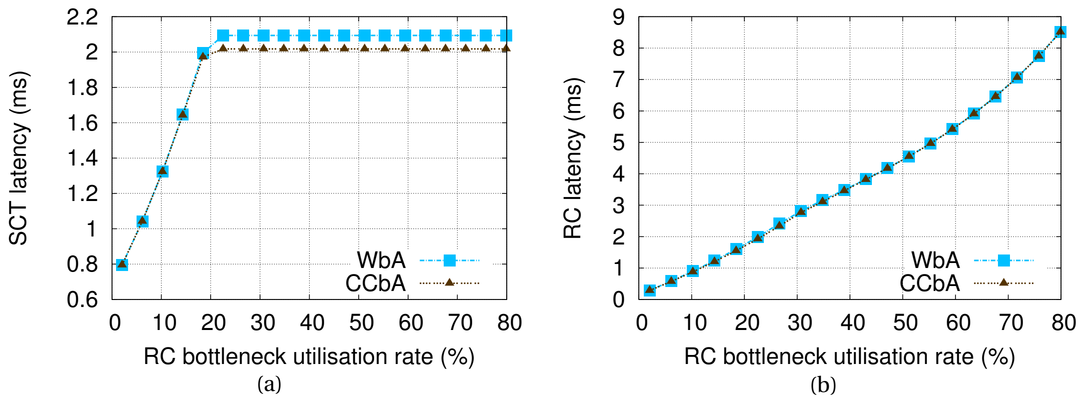


Figure 6.3: Comparing WbA and CCbA, impact of UR^{bn}_{RC} on: (a) SCT delay bound; (b) RC delay bound, with $Scenario_{RC} = \left(UR^{bn}_{SCT} = 20, UR^{bn}_{RC} \in [2:80], L_M = 22,118, L_R = 0, BW = 0.46\right)$

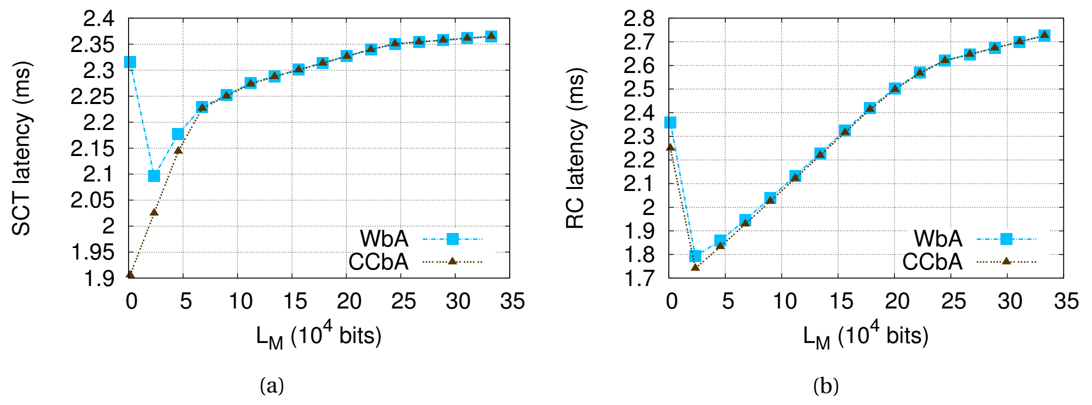


Figure 6.4: Comparing WbA and CCbA, impact of L_M on: (a) SCT delay bound; (b) RC delay bound, with $Scenario_{LM} = \left(UR^{bn}_{SCT} = 20, UR^{bn}_{RC} = 20.5, L_M \in [1382.4..216,830], L_R = 1177.6, BW = 0.46\right)$

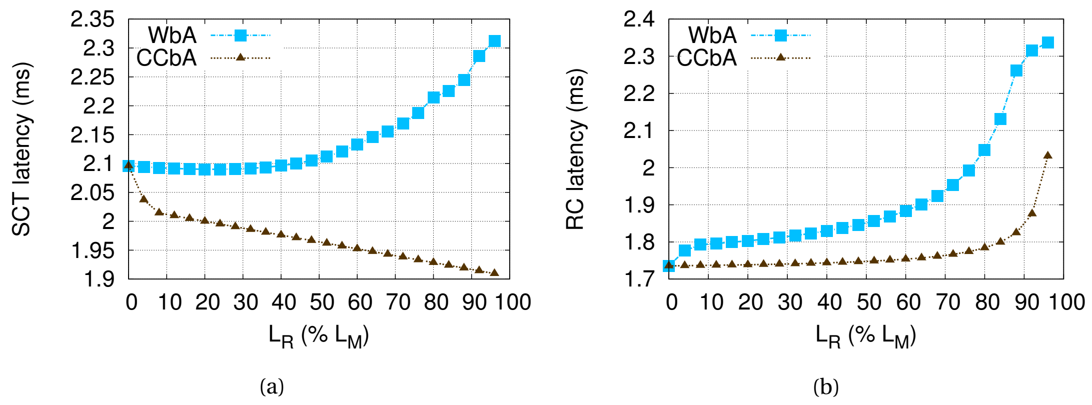


Figure 6.5: Comparing WbA and CCbA, impact of L_R on: (a) SCT delay bound; (b) RC delay bound, with $Scenario_{LR} = \left(UR^{bn}_{SCT} = 20, UR_{RC} = 20, L_M = 22,118, L_R \in [0..0.99] \cdot L_M, BW = 0.46\right)$

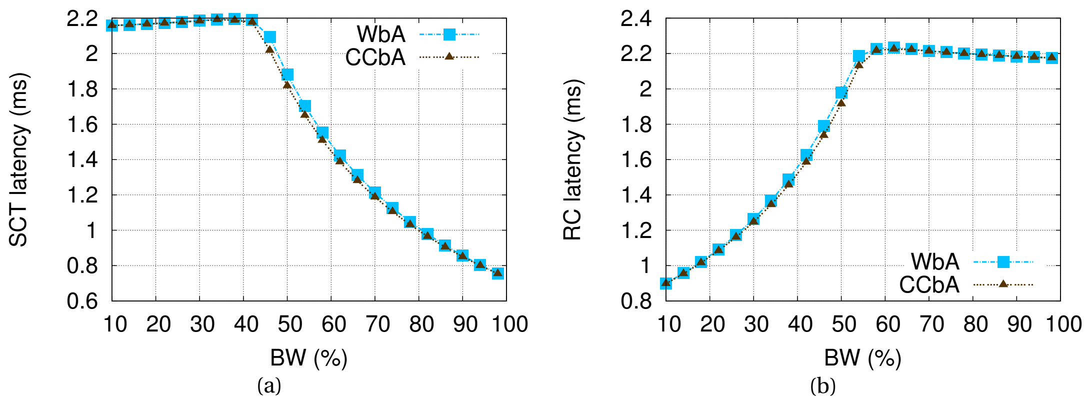


Figure 6.6: Comparing WbA and CCbA, impact of BW on: (a) SCT delay bound; (b) RC delay bound, with $Scenario_{BW} = \left(UR^{bn}_{SCT} = 20, UR^{bn}_{RC} = 20.5, L_M = 22,118, L_R = 1177.6, BW \in [0..0.99]\right)$

6.2.3 Comparing heuristic and dichotomous deadline tuning methods

In this section, we study the impact of optimised BLS parameters compared to intuitive parameters on a multi-hop network.

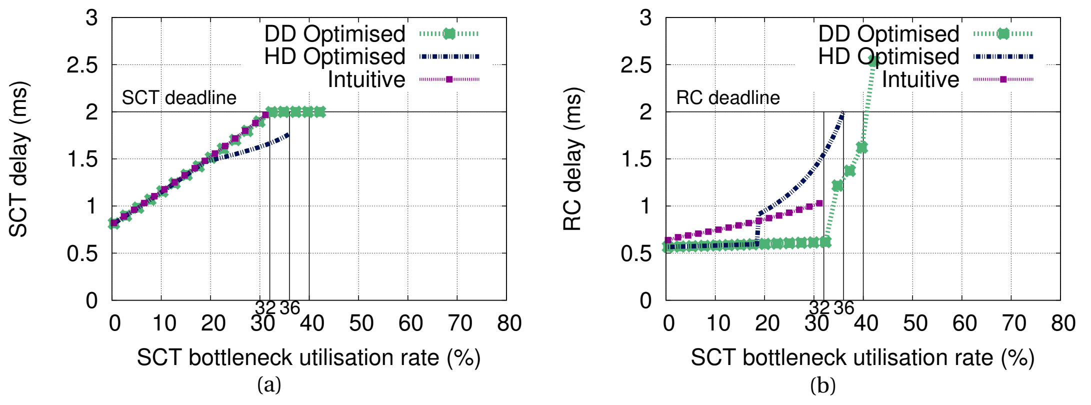


Figure 6.7: Comparing Intuitive and Optimised parameters, impact of UR^{bn}_{SCT} on: (a) SCT delay bound; (b) RC delay bound, with $Scenario_{SCT} = \left(UR^{bn}_{SCT} \in [0.4:79], UR^{bn}_{RC} = 20.5\right)$

In chapter 4, we have proposed two methods to compute the local deadline: the heuristic deadline (HD) and dichotomous deadline (DD) methods. Hence, we compare the delay bounds of SCT and RC under HD and DD methods, in reference to the intuitive parameterisation detailed in Section 5.3.5.

In Figures 6.7 and 6.8, the SCT and RC delay bounds are illustrated under $scenario_{SCT}$ and $scenario_{RC}$. It is worth noting that we only present the admissible results, i.e., when all the deadlines are fulfilled.

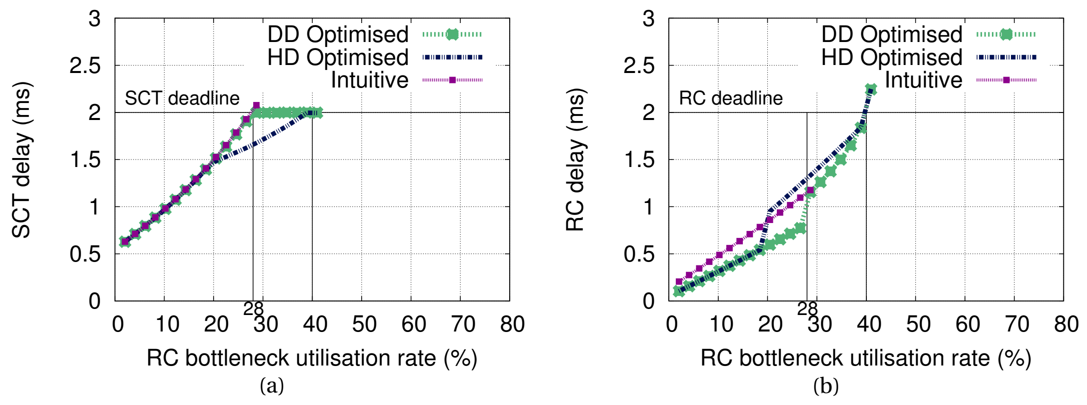


Figure 6.8: Comparing Intuitive and Optimised parameters, impact of UR_{RC}^{bn} on: (a) SCT delay bound; (b) RC delay bound, with $Scenario_{RC} = \left(UR_{SCT}^{bn} = 20, UR_{RC}^{bn} \in [2:80]\right)$

First, concerning the maximum bottleneck utilisation rates:

- in Figure 6.7, we note that the maximum bottleneck SCT utilisation rate is 32% with intuitive parameters, 36% with HD optimised parameters, and 40% with DD optimised parameters;

- in Figure 6.8, the maximum bottleneck RC utilisation rate is 28% with intuitive parameters, and 40% with both HD and DD tuning methods.

These results show an improvement of the SCT (resp. RC) schedulability up to 25% (resp. 42%) under the optimised parameters, in comparison to the intuitive ones.

Secondly, in Figure 6.7(b), the RC delay bounds with HD optimised parameters are lower than the delay bounds with intuitive parameters until UR_{SCT} = 18%. However, for UR_{SCT} between 18% and 32%, the intuitive tuning is better than the optimised tuning. The same issue is visible in Figure 6.8(b): for UR_{RC} between 20% and 28%, the intuitive parameters are better than the HD optimised ones.

To understand the reasons of this issue, in Figure 6.9, we present separately the SCT delay bound in the first and in the second switch output ports, denoted $delay_{SCT}^{mux1}$ and $delay_{SCT}^{mux2}$. In Figure 6.9(a), we can separate the optimised SCT delay bounds in 4 areas, with UR_{SCT}:

- between 0% and 20%, $delay_{SCT}^{mux1} < Deadline_{SCT}^{mux1}$ and $delay_{SCT}^{mux2} < Deadline_{SCT}^{mux2} \Rightarrow delay_{SCT}^{end2end} < Deadline_{SCT}^{end2end}$. The multiplexer deadline is reached in neither switches;

- between 20% and 45%, $delay_{SCT}^{mux1} = Deadline_{SCT}^{mux1}$, $delay_{SCT}^{mux2} < Deadline_{SCT}^{mux2} \Rightarrow delay_{SCT}^{end2end} = Deadline_{SCT}^{end2end}$. The switch output port 1 deadline (about 1ms, see Figure 6.9(a)) is reached and the SCT delay bound remains at this deadline $Deadline_{SCT}^{mux1}$ until UR_{SCT} = 50% (see Figure 6.9(a)). However, in the second switch output port, the SCT delay remains firmly below its deadline;

- between 45% and 50%, $delay_{SCT}^{mux1} = Deadline_{SCT}^{mux1}$, $delay_{SCT}^{mux2} = Deadline_{SCT}^{mux2} \Rightarrow delay_{SCT}^{end2end} = Deadline_{SCT}^{end2end}$. The SCT end-to-end delay bound is equal to the end-to-end deadline, as the delays in both output ports are equal to their respective deadlines (see Figure 6.9);

- between 50% and 80%, $delay_{SCT}^{mux1} > Deadline_{SCT}^{mux1}$ and $delay_{SCT}^{mux2} = Deadline_{SCT}^{mux2} \Rightarrow delay_{SCT}^{end2end} > Deadline_{SCT}^{end2end}$. The end-to-end delay bound is over the end-to-end deadline as the delay in the switch output port 1 is over its deadline (see Figure 6.9).

Hence, this highlights the fact that limiting the local deadline in an output port without taking into account the state of other ones in the path decreases the performance of the RC delay bounds.

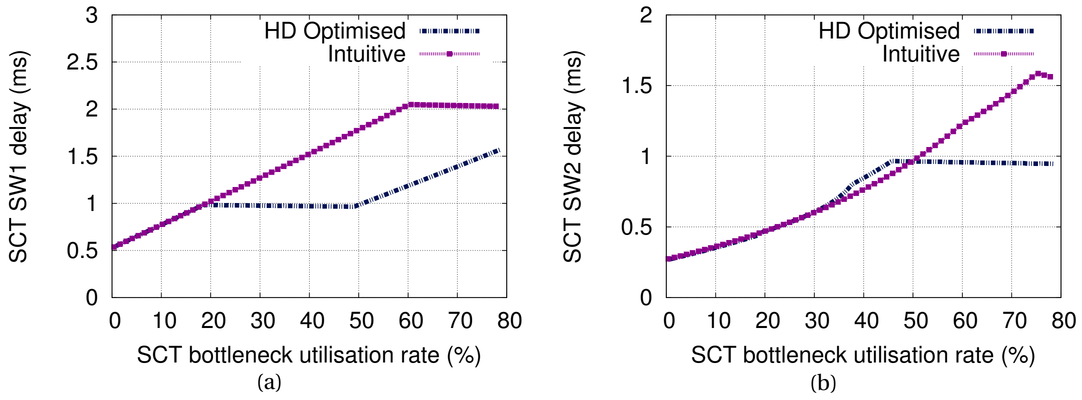


Figure 6.9: Intuitive vs HD method, impact of UR_{SCT}^{bn} on SCT delay bound in: (a) Switch 1 (b) Switch 2 with $Scenario_{SCT} = \left(UR_{SCT}^{bn} \in [0.4:79], UR_{RC}^{bn} = 20.5\right)$

Contrary to the HD method, the DD method takes into account the output ports along the flow path. In Figures 6.7 and 6.8, the RC delay bounds with DD optimised parameters is consistently better than the ones with both intuitive and HD optimised BLS parameters. For instance at $UR_{SCT} = 32\%$, the RC delay bound is improved by 49% with reference to intuitive parameters, and by 74% compared to HD optimised parameters.

We can conclude from this performance analysis that Dichotomous Deadline method leads to a great improvement over both the intuitive and Heuristic Deadline methods. The schedulability of SCT is actually increased by up to 31%, and the RC delay bound is decreased by up to 75%.

6.2.4 Comparative Performance Analysis with current and DRR-compliant AFDX under DD method

We compare extended, current and DRR-compliant AFDX when computing the parameters with the Dichotomous Deadline method (for both extended and DRR-compliant AFDX). The results are illustrated in Fig.6.10 and Fig.6.11.

First, concerning the maximum bottleneck utilisation rates:

- in Figure 6.10, we note that the maximum bottleneck SCT utilisation rate is 27% with the current AFDX, 35% with DRR-compliant AFDX and 41% with the extended AFDX.
- in Figure 6.11, the maximum bottleneck RC utilisation rate is 33% with the current AFDX, 38% with DRR-compliant and 41% with the extended AFDX.

Hence, the extended AFDX improves the maximum utilisation rate, compared to both the current AFDX (up to 24%) and DRR-compliant AFDX (up to 17%).

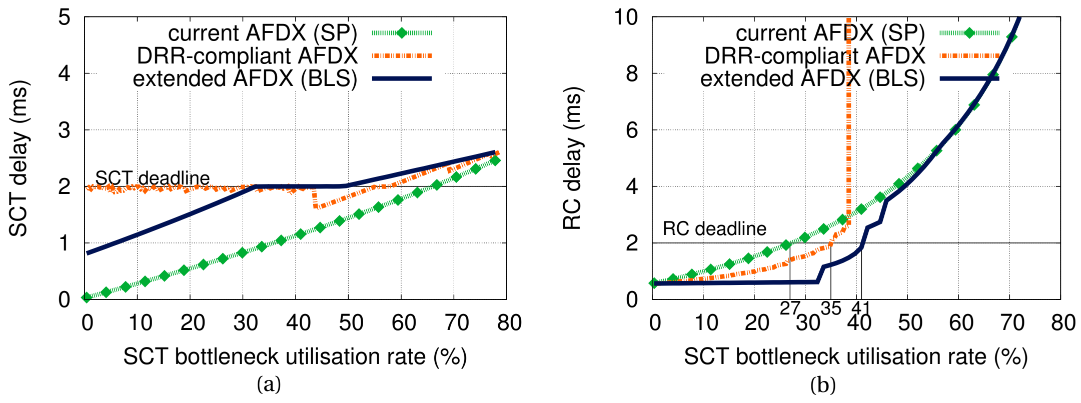


Figure 6.10: Dichotomous Deadline Optimisation method: impact of UR^{bn}_{SCT} on: (a) SCT delay bound; (b) RC delay bound with $Scenario_{SCT} = \left(UR^{bn}_{SCT} \in [0.4:79], UR^{bn}_{RC} = 20.5\right)$

Secondly, in Figure 6.10(b), the RC delay bounds with BLS are lower than the delay bounds with either DRR-compliant and current AFDX. The extended AFDX improves the RC delay bound up to 77% compared to the current AFDX, and up to 73% compared to DRR-compliant AFDX (when the SCT and RC deadlines are fulfilled). The same behaviour is visible in Figure 6.11(b): the extended AFDX improves the RC delay bounds up to 89% compared to the current AFDX, and up to 38% compared to DRR-compliant AFDX (when the deadlines are fulfilled).

The improvements of the RC delay bounds and schedulability with the extended (BLS) and DRR-compliant AFDX, in reference to the current AFDX (SP) are illustrated in Table 6.3. We have also computed the computation times with both tuning methods.

First, we can see that the DD method improves both the RC delay bounds and the maximum utilisation rates of SCT and RC, compared to the HD method. We note that the positive impact is much stronger under the extended AFDX than under the DRR-compliant AFDX.

However, the HD method is less complex, with a much lower computation time. For instance, in Table 6.3, we can see that the computation time is multiplied up to 52 times for DRR-compliant AFDX and up to 25 times for extended AFDX with the DD method, in reference to HD method.

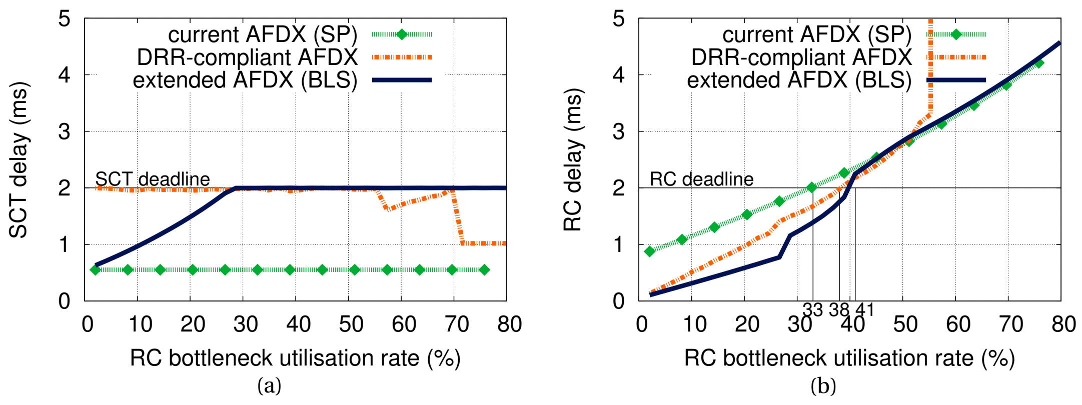


Figure 6.11: Dichotomous Deadline Optimisation method: impact of UR_{RC}^{bn} on: (a) SCT delay bound; (b) RC delay bound, with $Scenario_{RC} = \left(UR_{SCT}^{bn} = 20, UR_{RC}^{bn} \in [2:80]\right)$

Scheduler/Shaper		improvement compared to SP(%)				computation times (s)	
		maximum RC delay at		maximum		of Scenario	
		$UR_{SCT}^{bn} = 33\%$	$UR_{RC}^{bn} = 28\%$	UR_{SCT}^{bn}	UR_{RC}^{bn}	SCT	RC
HD method	BLS	18	22	33	21	57	9
	DRR	18	16	26	15	395	58
DD method	BLS	77	55	52	24	117	233
	DRR	18	17	30	15	15,000	3017

Table 6.3: Comparing parameter tuning methods

From these scenarios, we can conclude that with an accurate parameter tuning method, the extended AFDX implementing the BLS has a large positive impact on both SCT and RC, compared to the current AFDX implementing the SP scheduler or DRR-compliant AFDX. Both optimisation methods enhance the RC delay bounds and the schedulability of SCT and RC. Nevertheless, the dichotomous deadline method leads to better results with much higher computation times, in comparison to heuristic deadline method.

6.3 Avionics Application: adding A350 Flight Control to the AFDX

The flight control traffic is the most important on an aircraft. It is currently on private MIL-STD-1553B networks (which has been presented in Section 2.2) to keep it isolated from other traffics. In this part, we study the possibility of adding the Flight Control traffic to the AFDX using our proposed 3-classes extended AFDX.

6.3.1 Defining a new network architecture

The AFDX fulfils all the avionics requirements and can be used for Safety Critical traffic, such as flight Control traffic. However, the main concern is the way the Flight Control devices can be connected to the AFDX to guarantee the avionics requirements, particularly the safety rule stating that a single failure must not cause the loss of a function. After a reverse-engineering process of the current A350 flight control architecture illustrated in Fig.6.12, we have proposed a new architecture connecting the flight control calculators and actuators based on the extended AFDX technology, as illustrated in Fig.6.13.

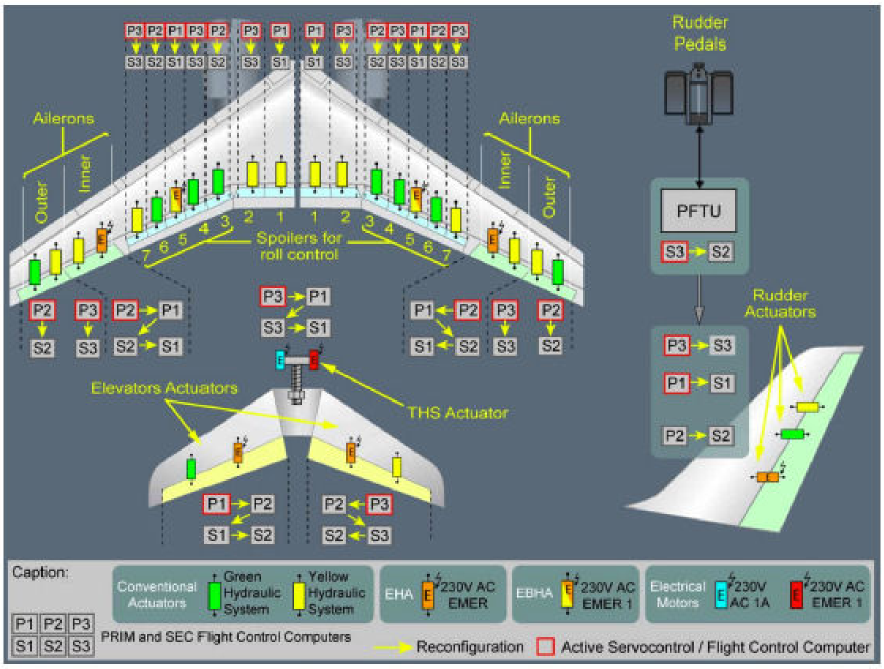


Figure 6.12: A350 flight control architecture, image downloaded from from https://www.quora.com/How-are-the-Airbus-A350-and-A330-different

First, to connect the calculators, we use 2 switches (SS1 and SS2) so that we do not lose a full network due to one switch loss. To obtain similar timing results for all actuators, we use the remaining 5 switches (L1, L2, C, R1, R2) to connect the actuators taking into account the energy supplier network, the type of actuator to fulfil the safety rule, and trying to have the same number of connections on each switch. The result is visible in Fig.6.13, we obtain a diamond-like structure with central switches (L1, L2, C, R1, R2) connected to 6 or 7 actuators.

In this architecture, the BLS is incorporated within the output ports from SS1/SS2 to the central switches, from the central switches to SS1/SS2, and from SS1/SS2 to the calculators. However, for the output port linking the central switches to the actuators, the BLS is not useful since the flight control traffic is the only type of traffic in these output ports. Hence, we obtain the network presented in Fig.6.14, where there is only one BLS along the path of each flow from a calculator to an actuator.

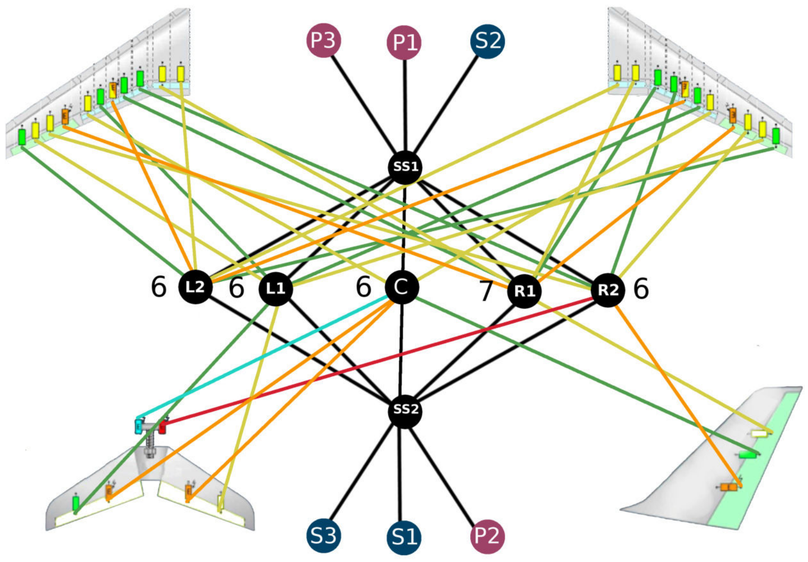


Figure 6.13: Primary network: new A350 flight control architecture

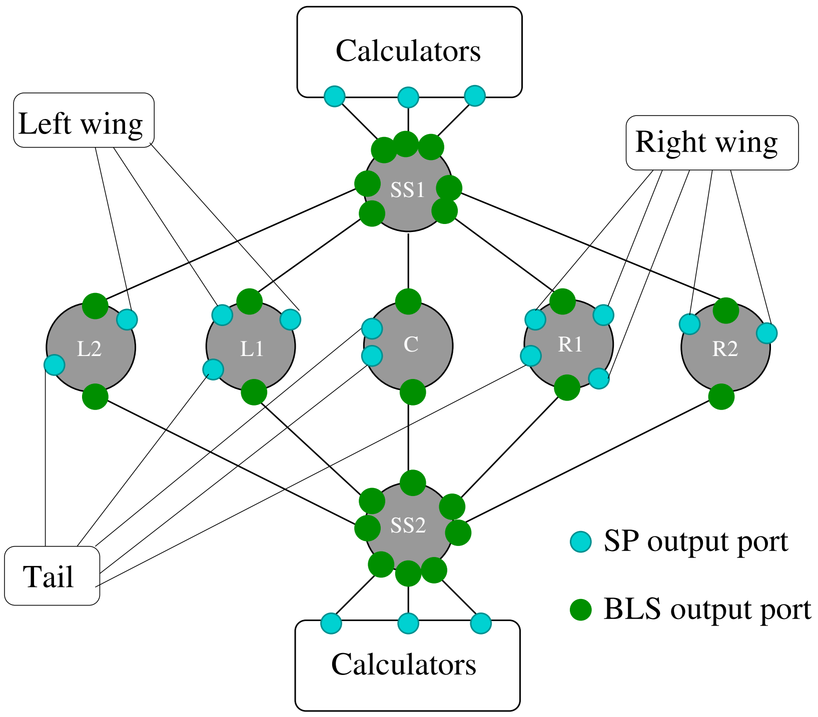


Figure 6.14: Output port type layout for the extended AFDX

6.3.2 Timing analysis of the proposed solution

In this section, we will consider many different scenarios to assess the performance of the extended AFDX solution in terms of delay bounds. The SCT traffic consists of Flight Control frames and the considered scenarios are described in Table 6.4.

We use the architecture presented in Fig.6.13. Hence, each calculator generates 17 flows with a periodicity BAG_{SCT} and a frame size MFS_{SCT}. Then, the frames arrive in SS1 or SS2. The output port with the heaviest load receives 7 flows from each of the 3 calculators connected to the considered switch; thus a total of 21 flows. Finally, the flows arrive in the central switches. Each central switch output ports receives 6 flows, one from each calculator.

The current heaviest load on the current 100Mbps AFDX is 30%. With the considered 1Gbps network, the heaviest load will be 3%. Hence, we consider that in each output port there are n_{RC} flows defined by MFS_{RC} and BAG_{RC} to obtain an utilisation rate of 3%: $n_{RC} = \left\lceil \frac{3}{100} \cdot C \cdot \frac{BAG_{RC}}{MFS_{RC}} \right\rceil$. Hence, we obtain $\alpha_{RC}(t) = n_{RC} \cdot \left(\frac{MFS_{RC}}{BAG_{RC}} \cdot t + MFS_{RC} \right)$.

We use this α_{RC} in both the calculators and the first switch (either SS1 or SS2). The delay bounds are computed with the CCbA modelisation and the BLS parameterisation are computed with the HD method. It is worth noting that since there is only one BLS along the path, the DD and HD methods are similar.

scenario	MFS_{SCT} (bytes)	BAG_{SCT} (ms)	$Deadline_{SCT}$ (ms)	MFS_{RC} (bytes)	BAG_{RC} (ms)	$Deadline_{RC}$ (ms)
1	64	2	1	320	2	2
2	64	1	1	320	2	2
3	64	1	0.5	320	2	2
4	64	1	1	320	4	4
5	64	1	1	640	2	2
6	64	1	1	1280	2	2
7	128	1	1	320	2	2
8	256	1	1	320	2	2

Table 6.4: Flight Control application: scenarios

The computed SCT and RC delay bounds under the different scenarios for the extended and current AFDX are detailed in Table 6.5. We consider two performance measurements. First, the end-to-end SCT delay bounds between a calculator and an end-system, denoted $delay_{SCT}^{end2end,m}$ for either the extended AFDX ($m = BLS$) or the current AFDX ($m = SP$). The goal is to verify that the end-to-end SCT deadline is fulfilled. Secondly, the RC delay bounds in the switches SS1 and SS2 under BLS. We denote $delay_{RC}^{SW1,m}$, the delay bound of the RC traffic in the first switch in the considered path, with $SW1$ being SS1 or SS2 depending on the considered flow, with $m \in \{BLS, SP\}$. It is worth noting that due to the symmetry of the network the delay bound in SS1 is identical to the one in SS2.

From the results in Table 6.5, we can see that for very diverse configurations, SCT delay bounds still fulfil the deadlines. Additionally, the RC delay bounds with the extended AFDX is always lower than the ones with the current AFDX, up to 49.9% of improvement in SW1. This confirms the efficiency of our proposal to handle mixed-criticality traffic.

scenario	$delay_{SCT}^{end2end,BLS}$ (μs)	$delay_{SCT}^{end2end,SP}$ (μs)	$delay_{RC}^{SW1,BLS}$ (μs)	$delay_{RC}^{SW1,SP}$ (μs)
1	120.00	56.17	44.89	55.16
2	121.96	58.15	44.89	55.63
3	121.96	58.15	44.89	55.63
4	184.54	60.09	84.73	95.92
5	129.91	66.01	47.46	58.25
6	145.80	81.74	52.60	63.48
7	176.53	112.36	45.55	67.76
8	298.62	233.71	46.79	93.38

Table 6.5: Flight Control application: results

6.4 Conclusion

In this chapter, we have validated the first conclusions of the preliminary analyses in previous chapters. First, we have used a generic avionics architecture to confirm the positive impact of the Continuous Credit based Approach in terms of schedulability and tightness, compared to Window based Approach.

Then, using the Continuous Credit based Approach, we have compared both parameter tuning methods, in comparison to intuitive method. Results show that both optimisation methods lead to better delay bounds, compared to intuitive parameters. Nevertheless, the dichotomous deadline optimisation improves schedulability (by up to 31%) and RC delay bounds (by up to 30%), but it induces much larger computation times (multiplied by up to 50 times), in comparison to the heuristic method. Hence, the choice of the method will depend on the complexity of the network, along side the time and computation power available. Furthermore, when comparing extended AFDX under both tuning methods to DRR-compliant and current AFDX, results have shown that with both methods, the extended AFDX implementing BLS has better schedulability and RC delay bounds than both current and DRR-compliant AFDX.

Finally, the validation of our proposal to support flight control traffic on the A350 using our extended AFDX has shown the large enhancement of the RC delay bounds, compared to the current AFDX implementing a SP scheduler (up to 50% of decrease), while guaranteeing the SCT schedulability.

CHAPTER
SEVEN

CONCLUSIONS AND PERSPECTIVES

"From the end spring new beginnings."

-Pliny the Elder

This chapter summarises the major contributions of this thesis and discusses the perspectives opened by our research.

The current AFDX backbone on the A380 and A350 is only used for essential systems, resulting in private networks for critical and non-essential traffics. The resulting architecture is very heterogeneous, leading to increased complexity, weight, delay and costs. With the strong increase of data exchanges and network complexity, the evolution of the current AFDX towards the next step of IMA architecture is inevitable. However, there are certain issues that we need to deal with before such a possibility can be realistic. Part of this issue comes from the industrial context: designing a new network is an expensive process, during the specification, development and certification. Hence, the rewards must be worth the costs. Concerning the research aspect of this problem, the challenge consists in finding the best solution specification and an accurate timing analysis for the latter.

In this thesis, we have proposed a solution to add mixed-criticality applications to the AFDX. We took industrial constraints into consideration, such as cost, weight, and reliability. These constraints lead to the conclusion that the best way to proceed is to modify the current AFDX. Due to the many existing ways to mix traffics, we started by identifying the main avionics requirements and challenges: predictability, modularity, complexity and fairness. After assessing the most relevant solutions, i.e., NP-SP, GPS, TTE, AVB, TSN, by comparing their benefits vs the identified requirements, we have identified the most promising solution: the Burst Limiting Shaper proposed by TSN, and the second best solution, DRR.

7.1 Conclusions

We summarise herein the main results of this thesis when following the methodology described in Chapter 1.

7.1.1 Specification

Concerning this first step, we have detailed in Chapter 3 the Burst Limiting Shaper (BLS) features and an eventual software implementation of this mechanism. Then, we have specified the full extended AFDX switch, incorporating the BLS. In particular, we have studied the possibilities offered by the unused field of an AFDX header for QoS identification. We have concluded that extending the current configuration files is the best solution: less complex, more scalable and with good performances. The comparison with the current AFDX has highlighted the few modifications needed at the software level, i.e., updating the static configuration table to manages additional priorities and at the hardware level, i.e., adding extra priority queues and implementing the BLS on top the NP-SP. We have concluded that the additional costs due to the switch modification and certification are counterbalanced by the homogenisation of the network, leading to decreased weight, cost and delays.

A preliminary analysis, conducted through *ns-2* simulations, has showed the good performances of the extended AFDX compared to the current AFDX, in terms of delay bounds and schedulability. However, simulations are not enough to prove that a system fulfils hard timing guarantees. This contribution has been published in [93].

7.1.2 Formal timing analysis

In Chapter 4, we proposed a formal timing analysis of the extended AFDX to formally prove that the hard real-time requirements are fulfilled. First, we have studied the existing work in this area. In particular, we have shown that the most recent one, based on the CPA method, is optimistic. Hence, a new modelisation is necessary. We have selected the Network Calculus framework, already used to certify the AFDX. We have first presented the timing analysis methodology. Then, we have detailed the modelisation of the BLS and then of the extended AFDX output port multiplexer. This has led to a discussion on the nature of the BLS showing the BLS is not a greedy shaper, and that it is better to consider the association of the BLS and NP-SP as a scheduler.

A sensitivity and tightness analyses have shown the good properties of our modelisation. Then, the comparison to the CPA model has shown that our model is less complex and has solved the optimistic and most of the pessimistic issues of the CPA. We have finished this chapter with a comparison of the extended AFDX to the current AFDX and a DRR-compliant AFDX (DRR being the second most promising mechanism). Results have shown that the best schedulability is obtained with the extended AFDX and that the RC delay bounds are noticeably enhanced with the extended AFDX.

Although we have obtained good results with this model, the evaluations have highlighted some scenarios where its accuracy could be improved. They have also shown the very large impact of the BLS parameters on the delay bounds; thus the importance of selecting appropriate ones to enhance performances. This contribution has been published in the case of three classes in [94].

7.1.3 Performance enhancement

In the first part of Chapter 5, we have proposed a second BLS modelisation, still using the Network Calculus framework. We start by identifying the flaw of the previous model, i.e., the continuity of the credit is not well-taken into account. Hence, this second modelisation is based on the continuity of the BLS credit. A sensitivity analysis has confirmed the good performances of this second model, compared to the first in terms of tightness. This contribution has been published in the case of three classes in [95].

Secondly, we have proposed a parameter tuning method to enhance the schedulability and delay bounds. We start by detailing the optimisation problem. Then, we propose resolution methods. One of the key step of the problem solving is to find good local deadlines within output port multiplexer. Thus, we have proposed two local deadline computation methods: one based on a heuristic, the other based on a dichotomous search. This contribution has been published in [96].

Our performance analysis has been conducted on a single hop network, giving the same local deadline with both methods. The comparison of the optimised vs intuitive parameters has shown the large schedulability and delay enhancements thanks to the optimisation. Finally, we have compared the extended AFDX to the current and DRR-compliant AFDX, leading to the conclusion that the extended AFDX enhances both the schedulability and delay bounds, compared to both current and DRR-compliant AFDX.

7.1.4 Validation

In a first part of Chapter 6, we have validated our proposal with a realistic multi-hop network. First, we have highlighted the good performances of the CCbA model over the WbA model. Then, we have compared the two optimisation methods to tune BLS parameters, i.e., heuristic deadline and dichotomous deadline, to intuitive parameters. Results have shown the beneficial aspects of the local deadline optimisation methods. However, in some situations, intuitive parameters provide better delay bounds than the heuristic deadline method. The second optimisation, dichotomous deadline method, solves this issue at the expense of the computation time. We use both methods to compare the performances of the extended AFDX to current and DRR-compliant AFDX. Results have shown that with both optimisation methods extended AFDX has the best performances, in terms of schedulability and delay bounds.

Finally, we conclude that both optimisation methods have merits. The heuristic deadline method gives adequate results with low computation time; whereas the dichotomous deadline method has better performances with much larger computation time. Hence, the selection of the tuning method depends on the computer power and complexity of the network and traffics.

In the second part of Chapter 6, we have considered a concrete avionics application: adding A350 flight control traffic to the AFDX. We have defined a new network architecture adapted to the flight control and we have computed the delay bounds using the heuristic deadline tuning method. We have considered multiple scenarios, by varying the size and period of the traffics. Across all the scenarios, we have shown the large enhancement of the delay bounds, compared to the current AFDX.

7.2 Perspectives

During this thesis, we have proposed an extension of the AFDX to support mixed-criticality applications. Our work has shown the positive impact of the Burst Limiting Shaper on the AFDX, using formal timing analyses on a 3-classes architecture. These results have opened new perspectives detailed in this section.

7.2.1 Testing other BLS configurations

The BLS is a very flexible scheduler, capable of supporting various configurations. In this thesis, we have selected an effective 3-classes configuration to limit the impact of the homogeneous SCT traffic on RC traffic, but the BLS can also be implemented for very diverse uses, such as heterogeneous avionics or internet traffics.

7.2.1.1 Application to heterogeneous avionics traffics

An interesting extension would be to consider more heterogeneous traffic classes and test the effect of using several BLS, for example using Fig.7.1.

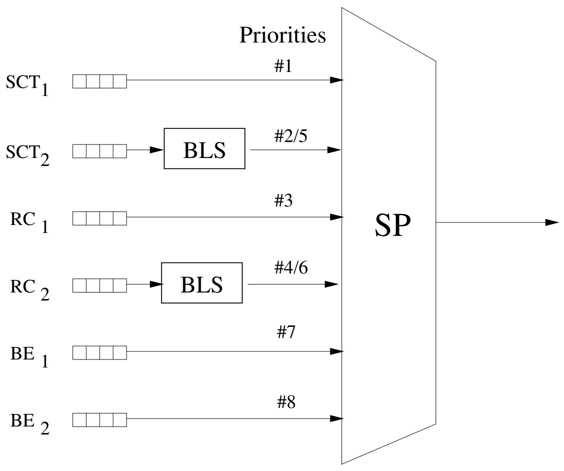


Figure 7.1: Example of a 6-classes output port of an extended AFDX switch

With this architecture, we can set the SCT (resp. RC) traffic with strict deadlines into the SCT_1 (resp. RC_1). SCT (resp. RC) traffics with less strict deadlines can be set into the SCT_2

(resp. RC_2) traffic classes. With this architecture, the BLS set for the SCT_2 queue can limit the impact of SCT_2 traffic on both RC and BE traffic classes, and the BLS set for the RC_2 queue can limit the impact of RC_2 traffic on BE traffic classes.

7.2.1.2 Application to Internet DiffServ core network architecture

An other application is to apply the BLS to the DiffServ core network architecture [84] [85]. The modelisation of the BLS done in Chapter 5 in Theorem 9 has highlighted the limitation of maximum utilisation rate available to the shaped class to share the bandwidth between several other classes. In the standard DiffServ core router architecture, illustrated in Fig.7.2, rate schedulers such as WRR, DRR, or WFQ are used to enforce the same sharing principle, to provide minimum service guarantees to the Assured Forwarding (AF) and Best effort (BE) classes. Thus, in [97], we have proposed to replace the standard rate scheduler schemes by a BLS, as illustrated in Fig.7.3. The idea is to leave to the real time traffic (UDP) the Efficient Forwarding class, with the first priority as is the case in Fig.7.2. The elastic traffic (TCP) in the AF is shaped by a BLS in order to make its output rate more predictable and less sensitive to EF traffic variations.

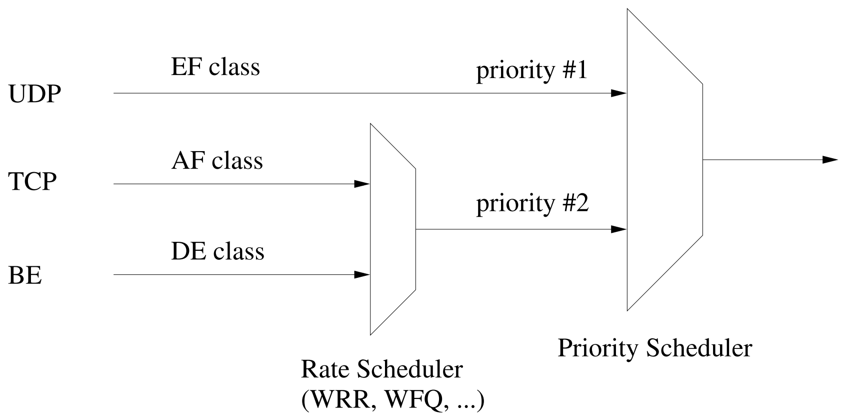


Figure 7.2: Current DiffServ core router architecture

Initial simulation results published in [97] have shown the good properties of the BLS to enforce more predictable output rates for the shaped class. We have presented our work to the ICCRG of the 99th IETF [98] under the name of Priority Switching Scheduler (encompassing both NP-SP and BLS) and published an Internet Draft [99] we currently defend as an independent submission validated by TSVWG. The next step is a real implementation of the architecture to test it on real internet traffic and on satellite links, to verify that the real behaviour fits the preliminary simulations. A second step is the implementation of several BLS to monitor their interactions.

7.2.2 Enhancing BLS parameter management

A second axis of perspectives concerns the BLS parameter management.

7.2.2.1 Generalising the parameter tuning

Throughout this thesis, we have highlighted the strong impact of BLS parameters on delay bounds. This has led to Section 5.3, where we have presented a parameter tuning for the 3-classes architecture. First, the computation of the delay bounds considered in the optimisation may be enhanced by considering alternative concatenation methods [100]. Secondly, at the start of Section 5.3, we have explained the difficulties posed by the generalisation of the parameter tuning, such as the issue of the interactions between multiple BLS. Nevertheless, to test other architecture configurations, a generalisation of the parameter tuning will be an interesting perspective.

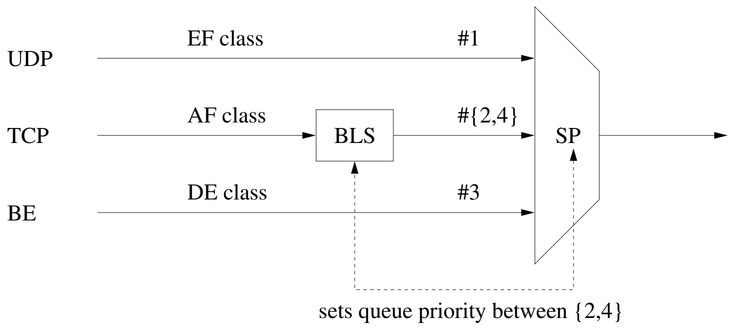


Figure 7.3: Proposed DiffServ core router architecture

7.2.2.2 Efficient network parameter setting

Over the last few years, there have been discussion about the best ways to set and update the parameters of a network. This have led to the development of the Software Defined Networks (SDN)[101]. SDN has been considered for TSN [102] and for the AFDX [103] but has yet to be implemented in an avionics network. We believe that using SDN to set and update the BLS parameters, when the aircraft is on the ground, could improve the management and maintenance of the AFDX network.

7.2.3 Considerations about the future of the AFDX

The commercial certification of TTE on board the 787-10 Dreamliner could soon make possible the integration of timing synchronisation within aircraft. In this case, our proposed extended AFDX can perfectly be combined with time-triggered mechanisms to mitigate the impact of the different classes on lower priorities. In particular, we have shown the benefits of using the BLS rather than the commonly used DRR to handle such an issue.

CHAPTER

EIGHT

APPENDIX

Contents

8.1 Computing Achievable Worst-Cases

We consider the 3-priority case study, where the SCT class is shaped by a BLS, presented in Section 3.4.1. As there is only one shaped class: SCT, we use $k = \emptyset$ to simplify, for the non-ambiguous notations, such as L_M^k or I_{send}^k. Our aim is to compute Achievable Worst-Cases for SCT and RC delays, i.e., realistic worst-cases.

In this section, we consider several hypothesis :

- traffics are packetized, i.e., we need to integrate the non-preemption impact;
- the same frame size for each class SCT, RC and BE, i.e., homogeneous traffic within each traffic class;

We use the four curves presented in Fig.8.1 and Fig.8.2 to compute two Achievable Worst-Cases for each traffic class, i.e., SCT and RC, for the single-hop network defined in Section 3.4.1. It is worth noting there is not strict order between the different cases. Based on these scenarios, we will be able to calculate so called Achievable Worst-Case delays to have an idea on the tightness of both RC and SCT delay bounds.

As illustrated in Fig.8.1 and Fig.8.2, there is an alternation of sending windows (when SCT traffic is sent) and idle windows (when RC traffic is sent). We call a *cycle* a sending window followed by an idle window (or an idle window followed by a sending window). A so called maximum-sized cycle is made of so called realistic maximum sending and idle windows: Δ_{send}^{real}, Δ_{idle}^{real}.

8.1.1 SCT achievable worst-cases

We start by presenting the methodology, before considering the two BLS behaviours described in Fig.8.1.

8.1.1.1 Methodology

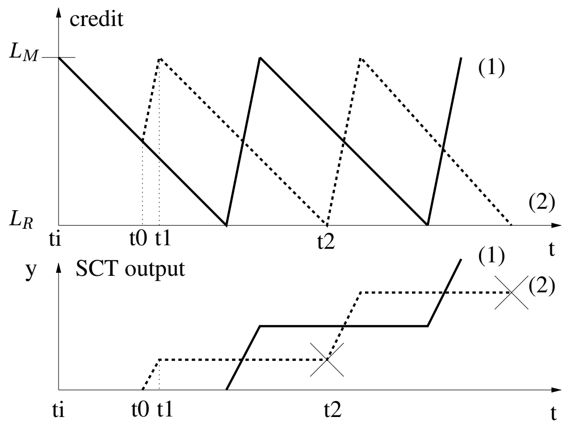


Figure 8.1: Two examples of worst-case BLS behaviours

To compute the worst-case delay of the SCT class, $delay_{SCT}^{max}$, we need to take into account of the following effects:

- BE class impact due to the non-preemption feature. We need to consider the transmission of a maximum-sized BE frame that may be transmitted before a SCT frame;

- Transmission time of SCT burst: it is the time needed for the output port multiplexer to transmit the maximum SCT burst $b_{SCT} = n^{in}_{SCT} \cdot MFS_{SCT}$, with a transmission capacity C, when taking into account the shaping effect of the upstream links. Each one of these link has a capacity C, resulting in the following transmission time:

$$\frac{n^{in}_{SCT} \cdot MFS_{SCT}}{C} - \frac{n^{in}_{SCT} \cdot MFS_{SCT}}{n^{links}_{SCT} \cdot C}$$

- RC class blocking effect $\Delta^{blocking}_{RC}$: it is the blocking effect of the shaper, which enforces the presence of $idle\ windows$ (resp. $sending\ windows$) to send the RC (resp. SCT) traffic.

Thus, we need to compute $\Delta^{blocking}_{RC}$ in each case to obtain the Achievable worst-cases.

The blocking effect depends the number of $realistic$ idle windows Δ^{real}_{idle} used by the RC traffic, denoted $Ncyle^{used}_{RC}$. The computation of $Ncyle^{used}_{RC}$ is based on both:

i) $Ncycle^{needed}_{RC}$, the number of cycles $needed$ to send the RC traffic during $realistic$ idle windows Δ^{real}_{idle};

ii) $Ncycle^{available}_{RC}$, the number of cycles $available$ to RC while the SCT traffic is being transmitted during $realistic$ sending windows Δ^{real}_{send}. If a sending window is started, it means that a full idle window Δ^{real}_{idle} is available to RC. We denote $Ncyle^{used}_{SCT}$ the number of Δ^{real}_{send} used by the SCT traffic. Hence, we have:

$$Ncycle^{available}_{RC} = \left\lceil Ncyle^{used}_{SCT} \right\rceil$$

Thus, we obtain:

$$Ncyle^{used}_{RC} = \min(Ncycle^{available}_{RC}, Ncycle^{needed}_{RC})$$

The computation of the number of cycles $Ncycle^{used}_{SCT}$ necessary to compute $\Delta^{blocking}_{RC}$ is based on:

i) the SCT and RC traffics;

ii) realistic windows, which depend on the chosen BLS behaviour and will be computed in each specific case.

So first, we assess the SCT and RC traffics. A strong hypothesis we make while computing the SCT traffic is that we do not consider the SCT traffic that may arrive while the SCT burst is being transmitted. Considering this additional traffic leads to the need of computing a fixed-point problem so we discard it here as it causes acceptable optimism rather than unacceptable pessimism. Hence, the maximum amount of considered SCT traffic is:

$$B^{max}_{SCT} = b_{SCT} = n^{in}_{SCT} \cdot MFS_{SCT}$$

However, not considering the amount of RC traffic arriving while SCT traffic is waiting leads to a large optimism. Hence, to compute the impact of RC, we need to compute the

the maximum amount of RC traffic that arrives while the SCT burst is being sent, i.e., during $delay_{SCT}^{max}$:

$$B_{RC}^{max}(delay_{SCT}^{max}) = n_{RC}^{in} \cdot MFS_{RC} \cdot (1 + \frac{delay_{SCT}^{max}}{BAG_{RC}})$$

Hence, the SCT delay is as follows:

$$delay_{SCT}^{max} = \frac{MFS_{BE}}{C} + \Delta_{RC}^{blocking}(delay_{SCT}^{max}) + \frac{n_{SCT}^{in} \cdot MFS_{SCT}}{C} - \frac{n_{SCT}^{in} \cdot MFS_{SCT}}{n_{SCT}^{links} \cdot C} \tag{8.1}$$

As $\Delta_{RC}^{blocking}(delay_{SCT}^{max})$ depends on $delay_{SCT}^{max}$, $delay_{SCT}^{max}$ can be computed by **solving this fixed point problem**. We consider $\frac{n_{SCT}^{in} \cdot MFS_{SCT}}{C}$ to be a good starting point.

8.1.1.2 SCT Achievable Worst-Case 1

We start by computing an Achievable Worst-Case for the SCT class, denoted SCT AWC-1 using the plain line curve (1) in Fig.8.1.

1. Computing the RC blocking delay $\Delta_{RC}^{blocking}(delay_{SCT}^{max})$
The RC blocking delay is defined by:

$$\Delta_{RC}^{blocking}(delay_{SCT}^{max}) = Ncycle_{RC}^{used} \cdot \Delta_{idle}^{real}$$

So, to compute the RC blocking delay, we need to compute the number of Δ_{idle}^{real} windows used by RC, $Ncycle_{RC}^{used}$.

To compute the number of cycles, we first need the realistic windows.

2. Computing realistic sending and idle windows
They are computed as the upper integer value of maximum number of frame that can be sent during a minimum window multiplied by the transmission time of a frame. Concerning the sending window, we consider that the window starts at $L_R^{min} = \max(0, L_R - I_{idle} \cdot \frac{MFS_{RC}}{C})$. This last hypothesis may be slightly optimistic as the window can in fact starts between L_R and L_R^{min} depending on the frames transmissions and sizes.

$$\Delta_{send}^{real} = \left\lceil \frac{\frac{L_M - L_R^{min}}{I_{send}}}{\frac{MFS_{SCT}}{C}} \right\rceil \cdot \frac{MFS_{SCT}}{C}$$

$$\Delta_{idle}^{real} = \left\lceil \frac{\frac{L_M - L_R}{I_{idle}}}{\frac{MFS_{RC}}{C}} \right\rceil \cdot \frac{MFS_{RC}}{C}$$

3. Computing $Ncycle_{RC}^{needed}$

We compute the number of cycles necessary to send the RC traffic $B_{RC}^{max}(delay_{SCT}^{max})$:

$$Ncycle_{RC}^{needed} = \frac{B_{RC}^{max}(delay_{SCT}^{max})}{C \cdot \Delta_{idle}^{real}}$$

4. Computing $Ncycle_{RC}^{available}$

Finally, the number of windows available to RC is:

$$Ncycle_{RC}^{available} = \left\lceil \frac{n_{SCT}^{in} \cdot MFS_{SCT}}{C \cdot \Delta_{send}^{real}} \right\rceil$$

8.1.1.3 SCT Achievable Worst-Case 2

For the second achievable worst-case, denoted SCT AWC-2, we use the dotted curve (2) in Fig.8.1.

1. Computing the RC blocking delay $\Delta_{RC}^{blocking}(delay_{SCT}^{max})$

To compute the RC blocking delay, we need to take into account the RC and SCT traffics sent between ti and t1. So, we compute:

i) the SCT traffic sent between t0 and t1;

ii) the RC traffic sent between ti and t0, and the corresponding window $\Delta_{tit0-idle}^{real}$.

Then, we can compute $Ncycle_{RC}^{used}$, the number of maximum-sized cycles, i.e., $\Delta_{send}^{real} + \Delta_{idle}^{real}$ used to send the RC burst remaining after $t1$, i.e., $B_{RC}^{max}(delay_{SCT}^{max}) - \Delta_{tit0-idle}^{real} \cdot C$.

Thus, we obtain the following RC blocking delay:

$$\Delta_{RC}^{blocking}(delay_{SCT}^{max}) = \Delta_{tit0-idle}^{real} + Ncycle_{RC}^{used} \cdot \Delta_{idle}^{real}$$

To compute the number of cycles, we first need the realistic windows between ti and t0, and between t0 and t1.

2. Computing realistic sending and idle windows

We consider that during the first idle window, there is no RC traffic backlogged when the credit reaches $\frac{L_M}{2}$. So SCT traffic is sent until there is again RC traffic, i.e., when the credit reaches L_R (see Fig.8.1).

First, we will use the same realistic maximum sending and idle windows as for AWC-1. We will also compute the realistic sending and idle windows from L_R to $\frac{L_M}{2}$ and from $\frac{L_M}{2}$ to L_R^{min}.

$$\Delta_{t0t1-send}^{real} = \left\lceil \frac{\frac{L_M}{2} - L_R^{min}}{I_{send}} \cdot \frac{C}{MFS_{SCT}} \right\rceil \cdot \frac{MFS_{SCT}}{C}$$

$$\Delta_{tit0-idle}^{real} = \left\lceil \frac{\frac{L_M}{2} - L_R}{I_{idle}} \cdot \frac{C}{MFS_{RC}} \right\rceil \cdot \frac{MFS_{RC}}{C}$$

3. Computing $Ncycle_{RC}^{needed}$

The number of maximum-sized cycles necessary to send the RC traffic after t1 is the numbers of cycles necessary to send the RC burst minus the traffic sent during $\Delta_{tit0-idle}^{real}$:

$$Ncycle_{RC}^{needed} = \frac{B_{RC}^{max}(delay_{SCT}^{max}) - \Delta_{tit0-idle}^{real} \cdot C}{C \cdot \Delta_{idle}^{real}}$$

4. Computing $Ncycle_{RC}^{available}$

To compute the number of cycle available to RC after t1, we consider the remaining SCT burst after t1. Thus, we have:

$$Ncycle_{RC}^{available} = \left\lceil \frac{n_{SCT}^{in} \cdot MFS_{SCT} - \Delta_{t0t1-send}^{real} \cdot C}{C \cdot \Delta_{send}^{real}} \right\rceil$$

8.1.2 RC achievable worst-cases

We start by presenting the methodology, before considering the two BLS behaviours described in Fig.8.2.

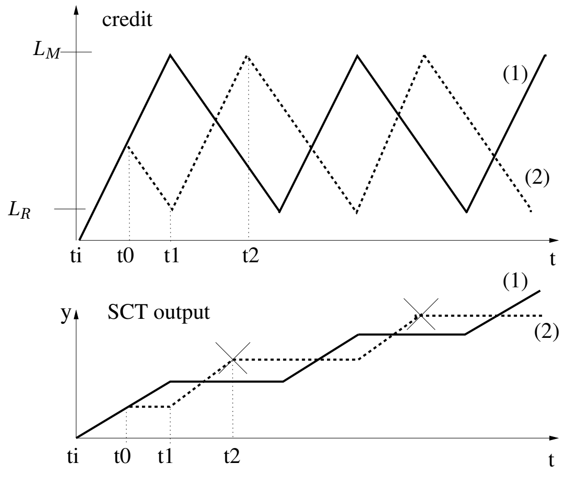


Figure 8.2: Two examples of best-case BLS behaviours

8.1.2.1 Methodology

To compute the worst-case delay of the RC class, $delay_{RC}^{max}$, we need to do an account of the following effects:

- BE class impact due to the non-preemption feature. We need to consider the transmission of a maximum-sized BE frame that may be transmitted before a RC frame;

- Transmission time of RC burst: it is the time needed for the output port multiplexer to transmit the maximum RC burst $b_{RC} = n_{RC}^{in} \cdot MFS_{RC}$, with a transmission capacity C, when taking into account the shaping effect of the upstream links. Each one of these link has a capacity C, resulting in the following transmission time:

$$\frac{n_{RC}^{in} \cdot MFS_{RC}}{C} - \frac{n_{RC}^{in} \cdot MFS_{RC}}{n_{RC}^{links} \cdot C}$$

- SCT class blocking effect $\Delta_{SCT}^{blocking}$: it is the blocking effect of the shaper, which enforces the presence of $idle\ windows$ (resp. $sending\ windows$) to send the RC (resp. SCT) traffic.

Thus, we need to compute $\Delta_{SCT}^{blocking}$ in each case to obtain the Achievable worst-cases.

The blocking effect depends the number of $realistic$ sending windows Δ_{send}^{real} used by the SCT traffic, denoted $Ncyle_{SCT}^{used}$. The computation of $Ncyle_{SCT}^{used}$ is based on both:

i) $Ncycle_{SCT}^{needed}$, the number of cycles $needed$ to send the SCT traffic during $realistic$ sending windows Δ_{send}^{real};

ii) $Ncycle_{SCT}^{available}$, the number of cycles $available$ to SCT while the RC traffic is being transmitted during $realistic$ idle windows Δ_{idle}^{real}. If an idle window is started, it means that a full sending window Δ_{send}^{real} is available to SCT. We denote $Ncyle_{RC}^{used}$ the number of Δ_{idle}^{real} used by the RC traffic. Hence, we have:

$$Ncycle_{SCT}^{available} = \left\lceil Ncyle_{RC}^{used} \right\rceil$$

Thus, we obtain:

$$Ncyle_{SCT}^{used} = \min(Ncycle_{SCT}^{available}, Ncycle_{SCT}^{needed})$$

The computation of the number of cycles $Ncycle_{SCT}^{used}$ necessary to compute $\Delta_{SCT}^{blocking}$ is based on:

i) the SCT and RC traffics;

ii) realistic windows, which depend on the chosen BLS behaviour and will be computed in each specific case.

So first, we assess the SCT and RC traffics. A strong hypothesis we make while computing the RC traffic is that we do not consider the RC traffic that may arrive while the RC burst is being transmitted. Considering this additional traffic leads to the need of computing a fixed-point problem so we discard it here as it causes acceptable optimism rather than unacceptable pessimism. Hence, the maximum amount of considered RC traffic is:

$$B_{RC}^{max} = b_{RC} = n_{RC}^{in} \cdot MFS_{RC}$$

However, not considering the amount of SCT traffic arriving while RC traffic is waiting leads to a large optimism. Hence, to compute the impact of SCT, we need to compute the

the maximum amount of SCT traffic that arrives while the RC burst is being sent, i.e., during $delay_{RC}^{max}$:

$$B_{SCT}^{max}(delay_{RC}^{max}) = n_{SCT}^{in} \cdot MFS_{SCT} \cdot (1 + \frac{delay_{RC}^{max}}{BAG_{SCT}})$$

Hence, the RC delay is as follows:

$$delay_{RC}^{max} = \frac{MFS_{BE}}{C} + \Delta_{SCT}^{blocking}(delay_{RC}^{max}) + \frac{n_{RC}^{in} \cdot MFS_{RC}}{C} - \frac{n_{RC}^{in} \cdot MFS_{RC}}{n_{RC}^{links} \cdot C} \tag{8.2}$$

As $\Delta_{SCT}^{blocking}(delay_{RC}^{max})$ depends on $delay_{RC}^{max}$, $delay_{RC}^{max}$ can be computed by **solving this fixed point problem**. We consider $\frac{n_{RC}^{in} \cdot MFS_{RC}}{C}$ to be a good starting point.

8.1.2.2 RC Achievable Worst-Case 1

We compute an Achievable Worst-Case for the RC class, denoted RC AWC-1 using the plain line curve (1) in Fig.8.2.

1. Computing the SCT blocking delay $\Delta_{SCT}^{blocking}(delay_{RC}^{max})$

An important difference between SCT and RC is the presence of an initial maximum sending windows starting at 0, denoted $\Delta_{send,0}^{real}$. It differs from the usual maximum sending windows which start at L_R. Hence, the initial SCT burst sent during this $\Delta_{send,0}^{real}$ must be taken into account throughout this computation of the SCT blocking delay.

To compute the SCT blocking delay, we need to compute:

i) $Ncycle_{SCT}^{available}$, the number of cycles available to SCT after t0, taking into account the impact of $\Delta_{send,0}^{real}$;

ii) $Ncycle_{SCT}^{needed}$, the number of cycles needed to send the SCT traffic remaining after t1, i.e., the SCT burst minus the traffic sent during $\Delta_{send,0}^{real}$.

Finally, we can compute $\Delta_{SCT}^{blocking}$, the interfering SCT traffic delay due to the shaper blocking effect:

$$\Delta_{SCT}^{blocking}(delay_{RC}^{max}) = \min\left(Ncycle_{SCT}^{needed}, Ncycle_{SCT}^{available}\right) \cdot \Delta_{send}^{real} + \Delta_{send,0}^{real}$$

To compute the number of cycles, we first need the realistic windows.

2. Computing realistic sending and idle windows

They are the same as Section 8.1.1, except for the fact that we consider for Δ_{send}^{real} that the window starts at L_R (instead of L_R^{min}) which may again be slightly optimistic as the window can in fact starts between L_R and L_R^{min} depending on the frames transmissions and sizes. Additionally, we consider the initial sending window $\Delta_{send,0}^{real}$.

$$\Delta_{send}^{real} = \left\lceil \frac{\frac{L_M - L_R}{I_{send}}}{\frac{MFS_{SCT}}{C}} \right\rceil \cdot \frac{MFS_{SCT}}{C}$$

$$\Delta_{idle}^{real} = \left\lceil \frac{\frac{L_M - L_R}{I_{idle}}}{\frac{MFS_{RC}}{C}} \right\rceil \cdot \frac{MFS_{RC}}{C}$$

$$\Delta_{send,0}^{real} = \left\lceil \frac{\frac{L_M - 0}{I_{send}}}{\frac{MFS_{SCT}}{C}} \right\rceil \cdot \frac{MFS_{SCT}}{C}$$

3. Computing $Ncycle_{SCT}^{needed}$

Next, we compute the number of window cycles necessary to send the SCT traffic remaining after t0. We must consider the first sending window, $\Delta_{send,0}^{real}$ which starts at 0:

The amount of traffic sent during this window is:

$$b_{SCT,0}^{max} = \Delta_{send,0}^{real} \cdot C$$

Finally, the number of cycles necessary to send the remaining SCT traffic after t0 is:

$$Ncycle_{SCT}^{needed} = \frac{B_{SCT}^{max}(delay_{RC}^{max}) - b_{SCT,0}^{max}}{C \cdot \Delta_{send}^{real}}$$

3. Computing $Ncycle_{SCT}^{available}$

The first available window is $\Delta_{send,0}^{real}$, and is taken into account directly in $\Delta_{SCT}^{blocking}(delay_{RC}^{max})$. So, We must remove 1 from $Ncycle_{RC}^{used}$. Finally, we have:

$$Ncycle_{SCT}^{available} = \left\lceil \frac{n_{RC}^{in} \cdot MFS_{RC}}{C \cdot \Delta_{idle}^{real}} \right\rceil - 1$$

8.1.2.3 RC Achievable Worst-Case 2

For the second achievable worst-case, denoted RC AWC-2, we use the dotted curve (2) in Fig.8.2.

1. Computing the SCT blocking delay $\Delta_{SCT}^{blocking}(delay_{RC}^{max})$

To compute the SCT blocking delay, we need to take into account the RC and SCT traffics sent between ti and t1. So, we compute:

i) the RC traffic sent between t0 and t1;

ii) the SCT traffic sent between ti and t0, and the corresponding window $\Delta_{tit0-send}^{real}$.

Then, we can compute $Ncycle_{SCT}^{used}$, the number of maximum-sized cycles, i.e., $\Delta_{send}^{real} + \Delta_{idle}^{real}$ used to send the SCT burst remaining after $t1$, i.e., $B_{SCT}^{max}(delay_{RC}^{max}) - \Delta_{t0t1-send}^{real} \cdot C$.

Thus, we obtain the following SCT blocking delay:

$$\Delta_{SCT}^{blocking}(delay_{RC}^{max}) = \Delta_{tit0-send}^{real} + Ncycle_{SCT} \cdot \Delta_{send}^{real}$$

To compute the number of cycles, we first need the realistic windows between ti and t0, and between t0 and t1.

2. Computing realistic sending and idle windows

We consider that during the first sending window, there is no SCT traffic backlogged when the credit reaches $\frac{L_M}{2}$. So RC traffic is sent until there is again SCT traffic, i.e., when the credit reaches L_R (see Fig.8.2).

First, we will use the same realistic maximum sending and idle windows as for AWC-1. We will also compute the realistic sending and idle windows from 0 to $\frac{L_M}{2}$ and from $\frac{L_M}{2}$ to L_R.

$$\Delta_{tit0-send}^{real} = \left\lceil \frac{\frac{L_M}{2} - 0}{I_{send}} \cdot \frac{C}{MFS_{SCT}} \right\rceil \cdot \frac{MFS_{SCT}}{C}$$

$$\Delta_{t0t1-idle}^{real} = \left\lceil \frac{\frac{L_M}{2} - L_R}{I_{idle}} \cdot \frac{C}{MFS_{RC}} \right\rceil \cdot \frac{MFS_{RC}}{C}$$

3. Computing $Ncycle_{SCT}^{needed}$

The number of maximum-sized cycles necessary to send the SCT traffic is the numbers of cycles necessary to send the SCT burst minus the traffic sent during $\Delta_{tit0-send}^{real}$:

$$Ncycle_{SCT}^{needed} = \frac{B_{SCT}^{max}(delay_{RC}^{max}) - \Delta_{tit0-send}^{real} \cdot C}{C \cdot \Delta_{send}^{real}}$$

4. Computing $Ncycle_{SCT}^{available}$

We consider the remaining RC burst after t1. Thus, we have:

$$Ncycle_{SCT}^{available} = \left\lceil \frac{n_{RC}^{in} \cdot MFS_{RC} - \Delta_{t0t1-idle}^{real} \cdot C}{C \cdot \Delta_{idle}^{real}} \right\rceil$$

8.2 Applying CPA model to the proposed architecture

In this section we apply CPA model [75] to the switch presented in Fig.3.1. CPA is based on an iterative approach. A local analysis is used to compute output event models maximising the transmission delay. The output event models then become the input event models of the next node. A global analysis loop is used to propagate event models. The analysis finishes if all models become stable, else the system is deemed unschedulable, e.g, if a constraint such as deadline, jitter, or delay is violated.

The model proposed in [75] necessitates at the local level solving an Integer Linear Program (ILP), a fixed-problem, and two maximisations. So, we compute the delay bounds for a single hop network. It will be a solid base of comparison between the different models.

8.2.1 SCT delay

The model separates the different queuing delays in several categories: the shaper blocking delay, denoted I_{SCT}^{SB}, the same priority blocking delay, denoted I_{SCT}^{SPB}, the lower priority delay, denoted I_{SCT}^{LPB} and the higher priority delay, denoted I_{SCT}^{HPB}. As SCT has the highest priority, there is no higher priority delay. The delays due to the fact that RC sometimes has a higher priory than SCT are taken into account in the shaper delay. In [75], the considered BLS low priority is the lowest available priority, as a result the impact of lower priority is taken into account in the shaper blocking delay, and not in I_{SCT}^{LPB}.

As we consider only one switch, we have three aggregate traffic flows: SCT, RC and BE flows, where each aggregate traffic flow is composed of n_k^{in} identical flows of class k, defined by a maximum frame size $MFS_k$and a period BAG_k. We call n_{SCT}^{links} the number of input links with SCT traffic (all have a capacity C).

First, we compute the delay for the q-th arrival of a SCT frame which arrives at a_{SCT}^q. We start by computing the blocking effects:

- $I_{SCT}^{LPB} = 0$
- $I_{SCT}^{SPB} = (q-1) \cdot \frac{MFS_{SCT}}{C}$
- $I_{SCT}^{SB} = \left\lceil \frac{q \cdot \frac{MFS_{SCT}}{C}}{t_{SCT}^{S-}} \right\rceil \cdot t_{SCT}^{R+}$

with:

- $t_{SCT}^{R+} = \lceil \frac{L_M - L_R}{I_{idle}} \rceil + \max_{j \in \{RC,BE\}} \frac{MFS_j}{C}$, the maximum replenishment interval
- $t_{SCT}^{S-} = \max\left\{ \left\lfloor \frac{L_M - L_R}{I_{send}} \right\rfloor, \max_{j \in SCT} \frac{MFS_j}{C} \right\}$, the shortest SCT service interval

Hence, we obtain the full transmission delay by adding the transmission time of a maximum sized frame to the queueing delays $\frac{MFS_{SCT}}{C}$:

$$delay_{SCT}^{max}(q) = q \cdot \frac{MFS_{SCT}}{C} + \left\lceil \frac{q \cdot \frac{MFS_{SCT}}{C}}{t_{SCT}^{S-}} \right\rceil \cdot t_{SCT}^{R+} - a_{SCT}^q$$

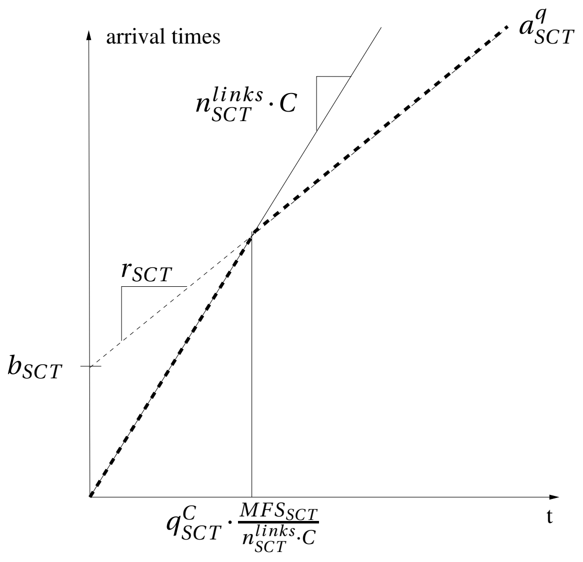
The largest worst-case delay of frame q is obtained by testing all the set of arrival candidate a_{SCT}^q as explained in [75]. In the case of SCT, its worst-case is obtained when considering the shortest arrival duration a_{SCT}^q,which can be computed as follows.

To compute a_{SCT}^q, we propose to take into account two behaviours as illustrated in Fig.8.3. First, the fact that frames can arrive at a maximum rate C from n_{SCT}^{links} due to traffic bursts. Secondly, a flow is characterised by its burst $b_{SCT} = n_{SCT}^{in} \cdot MFS_{SCT}$ and input rate $r_{SCT} = \frac{b_{SCT}}{BAG_{SCT}}$.

Hence, the shortest arrival time of a frame can be separated in two part: i) the number of frames that can arrive with a rate $n_{SCT}^{links} \cdot C$ multiplied by their arrival times; ii) the remaining frames multiplied by their arrival times. This gives:

$$a_{SCT}^q = \min(q, q_{SCT}^C) \cdot \frac{MFS_{SCT}}{n_{SCT}^{links} \cdot C} + (q - q_{SCT}^C)^+ \cdot \frac{MFS_{SCT}}{r_{SCT}}$$

with the maximum number of frames that can be sent at rate C, denoted q_{SCT}^C.



arrival times
a_{SCT}^{q}
$n_{SCT}^{links} \cdot C$
r_{SCT}
b_{SCT}
$q_{SCT}^{C} \cdot \frac{MFS_{SCT}}{n_{SCT}^{links} \cdot C}$
t

Figure 8.3: Computing a_{SCT}^{q}

Using the intersecting functions illustrated Fig.8.3, we can compute the intersection time: $q_{SCT}^{C} \cdot \frac{MFS_{SCT}}{n_{SCT}^{links} \cdot C} = \frac{b_{SCT}}{n_{SCT}^{links} \cdot C - r_{SCT}}$

Finally, we take into account the fact that $q_{SCT}^{C} \in \mathbb{N}$. Hence, q_{SCT}^{C} is such as:

$$q_{SCT}^{C} = \left\lfloor n_{SCT}^{in} \cdot \frac{n_{SCT}^{links} \cdot C}{n_{SCT}^{links} \cdot C - r_{SCT}} \right\rfloor$$

The worst-case transmission of a SCT frame is obtained by finding the maximum delay for q varying from 1 to q_{max}, the maximum number of frames in the longest SCT-busy period as explained in [104]. For each flow of class k, the maximum number of frames arriving in δ is $\left\lceil \frac{\delta}{BAG_k} \right\rceil$. So, when considering n_{SCT}^{in} flows we obtain:

$$q_{SCT}^{max} = \left\lceil n_{SCT}^{in} \cdot \frac{delay_{SCT}^{max}(q)}{BAG_{SCT}} \right\rceil$$

This q_{SCT}^{max} can be computed after $delay_{SCT}^{max}$ to make sure whether the condition $q \leqslant q_{SCT}^{max}$ if fulfilled and finish the computation if such is not the case.

8.2.2 RC delay

For the RC delay, we have the same composition of the delay, but this time there are higher priority traffics: the SCT class. As SCT is a shaped class, we only have to consider $I_{RC}^{HPB,BLS}$, and not $I_{RC}^{HPB,nBLS}$ which applies to non shaped higher priority traffic. As RC is not shaped, the shaper blocking delay is null. First, we compute the delay for the q-th arrival of a RC frame which arrives at a_{RC}^{q}. We start by computing the blocking effects:

$$I_{RC}^{SB} = 0,$$

and the lower priority impact is taken into account in I_{SCT}^{LPB}:

$$I_{RC}^{LPB} = \frac{MFS_{BE}}{C}.$$

The impact of same priority is similar to SCT:

$$I_{RC}^{SPB} = (q-1) \cdot \frac{MFS_{RC}}{C}.$$

The computation of $I_{RC}^{HPB,BLS}$ however is much more complicated. We consider only one higher priority flow: the aggregate traffic of class SCT. Depending on a time interval δ, we must compute x_{SCT}^{max}, the maximum number of interfering SCT frames during δ, denoted $x_{SCT}(\delta)$, is:

$$I_{RC}^{HPB,BLS}(\delta) = x_{SCT}^{max} \cdot \frac{MFS_{SCT}}{C} = \max(x_{SCT}(\delta)) \cdot \frac{MFS_{SCT}}{C}$$

The computation of x_{SCT}^{max} is an ILP problem. The workload problem has been formulated in Eq.(9) in [75]. The value of $x_{SCT}(\delta)$ is of course linked to the considered δ, but also to a credit replenishment: when a class different from SCT sends frames, the credit decreases (is replenished). We call $x_{SCT}^{R}(\delta)$ the number of replenishing intervals. As SCT is the only class with a priority higher than RC, the second term of Eq.(9) from [75], x_j^D, is null. Additionally, $x_{SCT}(\delta)$ and $x_{SCT}^{R}(\delta)$ are constrained by the credit as defined in Eq.(8) in [75]. So, Eq.(8) and Eq.(9) from [75] give the following two inequations:

$$0 \leqslant x_{SCT}(\delta) \cdot I_{send} \cdot \frac{MFS_{SCT}}{C} - x_{SCT}^{R}(\delta) \cdot t_{SCT}^{R+} \cdot I_{idle} \leqslant L_M + \frac{MFS_{SCT}}{C} \cdot I_{send}$$

$$0 \leqslant x_{SCT}(\delta) \cdot \frac{MFS_{SCT}}{C} + x_{SCT}^{R}(\delta) \cdot t_{SCT}^{R-} \leqslant \delta + \frac{MFS_{SCT}}{C}$$

with $t_{SCT}^{R-} = \left\lfloor \frac{L_M - L_R}{I_{idle}} \right\rfloor$ the minimum replenishment interval, and with the following constraints:

- $x_{SCT}(\delta)$ is upper bounded by the number of frames that can arrive during a right half-opened interval δ, i.e., the maximum burst plus the frames arriving during δ at the flow rate;
- $x_{SCT}^{R}(\delta)$ is the number of replenishment intervals, it is upper bounded by considering the shortest service interval of class SCT.

This gives:

$$0 \leqslant x_{SCT}(\delta) \cdot \frac{MFS_{SCT}}{C} \leqslant n_{SCT}^{in} \cdot MFS_{SCT} \cdot \left(\frac{\delta}{BAG_{SCT}} + 1\right),$$

$$0 \leqslant x_{SCT}^{R}(\delta) \leqslant \left\lceil \frac{\delta}{t_{SCT}^{S-} + t_{SCT}^{R-}} \right\rceil$$

with: $t_{SCT}^{S-} = \max\left(\left\lfloor \frac{L_M - L_R}{I_{send}} \right\rfloor, \frac{MFS_{SCT}}{C}\right)$, $t_{SCT}^{R-} = \left\lfloor \frac{L_M - L_R}{I_{idle}} \right\rfloor$, and $t_{SCT}^{R+} = \left\lceil \frac{L_M - L_R}{I_{idle}} \right\rceil + \max_{j \in RC, BE} \frac{MFS_j}{C}$.

So, we have defined the ILP problem depending on δ. This δ is actually the variable of the fixed-point problem defined in Eq.(8.4).

Next, we can compute the maximum queuing delay $QD_{RC}^{max}(q)$.

$$QD_{RC}^{max}(q) = \frac{MFS_{BE}}{C} + (q-1)\cdot\frac{MFS_{RC}}{C} + I_{RC}^{HPB,BLS}(QD_{RC}^{max}(q)) \tag{8.3}$$

By implementing both the fixed point problem and the ILP problem we can compute the RC queuing delay depending on q

The largest transmission delay, denoted $delay_{RC}^{max}$, can be computed by adding the transmission time and removing the arrival delay a_{RC}^{q} from the maximum queuing delay $QD_{RC}^{max}(q)$.

$$delay_{RC}^{max}(q) = QD_{RC}^{max}(q) + \frac{MFS_{RC}}{C} - a_{RC}^{q} \tag{8.4}$$

The largest worst-case delay of frame q is obtained by testing all the set of arrival candidates a_{RC}^{q} as explained in [75]. In the case of RC, its worst-case is obtained when considering the shortest arrival duration a_{RC}^{q}, which we propose to compute similarly to SCT.

$$a_{RC}^{q} = \min(q, q_{RC}^{C})\cdot\frac{MFS_{RC}}{n_{RC}^{links}\cdot C} + (q - q_{RC}^{C})^{+}\cdot\frac{MFS_{RC}}{r_{RC}}$$

with the maximum number of frames that can be sent at rate C:

$$q_{RC}^{C} = \left\lfloor n_{RC}^{in}\cdot\frac{n_{RC}^{links}\cdot C}{n_{RC}^{links}\cdot C - r_{RC}}\right\rfloor$$

Finally, similarly to SCT, the worst-case transmission of a RC frame is obtained by finding the maximum delay for q varying from 1 to q_{RC}^{max}, the maximum number of frames in the longest RC-busy period [104]:

$$q_{RC}^{max} = \left\lceil n_{RC}^{in}\cdot\frac{delay_{RC}^{max}(q)}{BAG_{RC}}\right\rceil$$

This q_{RC}^{max} can be computed after $delay_{RC}^{max}$ to make sure whether the condition $q \leqslant q_{RC}^{max}$ if fulfilled and finish the computation if such is not the case.

8.3 Window-based Approach model proofs

The windows we use for the model are highlighted in Fig.8.4. In both proofs, to compute the idle and sending windows, we consider only class k traffic and MC(k) flows because:

- for the strict minimum service curve, the impact of $LC(k)$ is taken into account in $\beta_{k\in BLS,p_H(k)}^{sp}$;

- for the strict minimum service curve, the impact of $HC(k)$ is taken into account in the residual minimum service curve offered to class k by $HC(k)$;
- for the maximum service curve, to compute the best-case, we consider that neither $HC(k)$ nor $LC(k)$ interfere with class k.

In both proofs, consider $R^*_j(t)$ the output traffic cumulative function of a class or set of classes $j \in \{k, MC(k)\}$, and $\Delta R^*_j(\delta)$ the variation of the output cumulative function during δ.

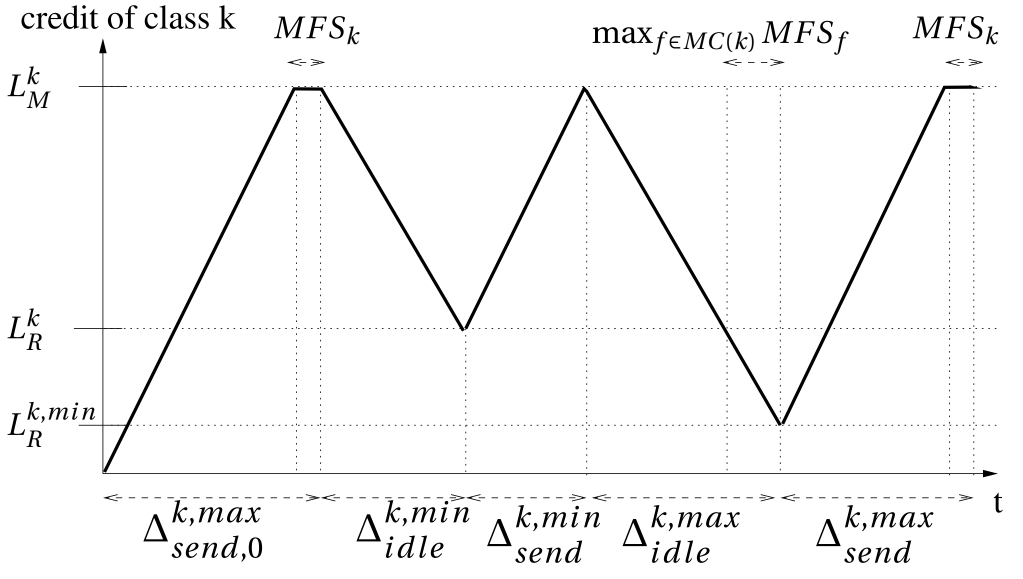


Figure 8.4: Idle and sending windows of a class k

We start by the strict minimum service curve proof.

8.3.1 Th.4: WbA strict minimum service curve

Consider a backlogged period for the class k, and a server with a strict minimum service curve $\beta(t)$ left by HC(k), which computation is based on Corollary 1.

During δ, the class k traffic can send a minimum cumulative amount of traffic. This amount can be described using the minimum sending window: $\exists p \in R^+$ such as:

$$\Delta R^*_k(\delta) \geq p \cdot \beta(\Delta^{k,min}_{send}) \tag{8.5}$$

The idea is to find a lower bound of p to define the service curve guaranteed to k, β^{bls}_k.

During $p \cdot \Delta^{k,min}_{send}$, the amount of consumed credit is $p \cdot (L^k_M - L^k_R)$. Additionally, in the worst-case, the credit started at its maximum value L^k_M, leading to an initial credit deficit of $L^k_M - L^k_R$. Hence, at least $(p+1) \cdot (L^k_M - L^k_R)$ credit must be gained during δ. The worst-case for the class k occurs if credit is gained by sending MC(k) frames in the maximum amount of time: $(p+1) \cdot \Delta^{k,max}_{idle}$. We obtain the following upper bound for $\Delta R^*_{MC(k)}(\delta)$:

$$\Delta R^*_{MC(k)}(\delta) \le (p+1)\cdot\beta(\Delta^{k,max}_{idle}) \tag{8.6}$$

Giving the strict minimum service curve property of $\beta(t)$ and using Eq.(8.6), we have:

$$\begin{aligned}\beta(\delta) &\leqslant \Delta R^*_k(\delta) + \Delta R^*_{MC(k)}(\delta)\\ &\leqslant \Delta R^*_k(\delta) + (p+1)\cdot\beta(\Delta^{k,max}_{idle})\end{aligned}$$

Consequently, the lower bound of p is as follows:

$$p \;\geqslant\; \frac{\beta(\delta)-\Delta R^*_k(\delta)}{\beta(\Delta^{k,max}_{idle})} - 1 \tag{8.7}$$

When injecting Eq.(8.7) in Eq.(8.5), we obtain:

$$\begin{aligned}
\Delta R^*_k(\delta) &\geqslant \left(\frac{\beta(\delta)-\Delta R^*_k(\delta)}{\beta(\Delta^{k,max}_{idle})} - 1\right)\cdot\beta(\Delta^{k,min}_{send})\\
\Delta R^*_k(\delta)\cdot\left(1+\frac{\beta(\Delta^{k,min}_{send})}{\beta(\Delta^{k,max}_{idle})}\right) &\ge \left(\frac{\beta(\delta)}{\beta(\Delta^{k,max}_{idle})}-1\right)\cdot\beta(\Delta^{k,min}_{send})\\
\Delta R^*_k(\delta) &\geqslant \frac{\frac{\beta(\delta)}{\beta(\Delta^{k,max}_{idle})}-1}{\frac{\beta(\Delta^{k,min}_{send})}{\beta(\Delta^{k,max}_{idle})}+1}\cdot\beta(\Delta^{k,min}_{send})
\end{aligned}$$

Given that $\Delta R^*_k(\delta) \ge 0$, then:

$$\Delta R^*_k(\delta) \geqslant \frac{\beta(\Delta^{k,min}_{send})}{\beta(\Delta^{k,min}_{send})+\beta(\Delta^{k,max}_{idle})}\cdot\left(\beta(\delta)-\beta(\Delta^{k,max}_{idle})\right)^+$$

Finally, as we consider a server with a constant rate C, the strict minimum residual service left by HC(k) is:

$$\beta(t) = C\cdot t - \sum_{j\in HC(k)}\alpha_j(t) = \left(C-\sum_{j\in HC(k)} r_j\right)\cdot t - \sum_{j\in HC(k)} b_j$$

which gives:

$$\Delta R^*_k(\delta) \ge \frac{\beta(\Delta^{k,min}_{send})}{\beta(\Delta^{k,min}_{send})+\beta(\Delta^{k,max}_{idle})}\cdot\left(C-\sum_{j\in HC(k)} r_j\right)\cdot\left(\delta-\Delta^{k,max}_{idle}\right)^+$$

8.3.2 Th.5: WbA maximum service curve

First, it is obvious that $\gamma(t) = C\cdot t$ is always a maximum service curve for class k.

Secondly, we compute $\gamma(t)$ when MC(k) traffic is enqueued.

From [87], we know that $\Delta R_k^*(t-s) \leqslant B(s)+\gamma(t-s)$, with $B(s)$ the backlog at s. We search for z such as $\Delta R_k^*(t-s) \leqslant z$. As $B(s) \geqslant 0$, we obtain:

$$\Delta R_k^*(t-s) \leqslant z \leqslant B(s)+z$$

Hence, we select $\gamma(t-s)=z$, which gives:

$$\Delta R_k^*(t-s) \leqslant \gamma(t-s)$$

During δ, the class k traffic can send a maximum cumulative amount of traffic. This amount can be described using the sending windows: $\forall \delta \in R^+$, $\exists p \in R^+$ such as:

$$\Delta R_k^*(\delta) \leqslant p \cdot \gamma(\Delta_{send}^{k,max}) + \gamma(\Delta_{send,0}^{k,max}) \tag{8.8}$$

During $\Delta_{send}^{k,max}$, the credit consumed is $L_M^k - L_R^{k,min}$ and during $\Delta_{send,0}^{k,max}$, the consumed credit is L_M^k. Hence, during δ:

$$credit_{consumed}^k \leqslant p \cdot \left(L_M^k - L_R^{k,min}\right) + L_M^k$$

Since the credit is a continuous function between 0 and L_M^k, the credit variation (gained credit plus consumed credit) is comprised between L_M^k and $-L_M^k$. Thus, we have:

$$\begin{aligned}
credit_{gained}^k + credit_{consumed}^k &\leqslant L_M^k \\
credit_{gained}^k &\leqslant -credit_{consumed} + L_M^k \\
credit_{gained}^k &\leqslant -p \cdot \left(L_M^k - L_R^{k,min}\right) \leqslant -p \cdot \left(L_M^k - L_R^k\right)
\end{aligned}$$

As gaining credit is a negative credit variation, $credit_{gained}^k \leqslant 0$. Thus:

$$\left|credit_{gained}^k\right| \geqslant p \cdot \left(L_M^k - L_R^k\right)$$

Additionally, the best-case for the class k occurs if credit is gained by sending MC(k) frames in the minimum amount of time. Since $p \cdot \Delta_{idle}^{k,min}$ is the minimum amount of time necessary to regain $p \cdot \left(L_M^k - L_R^k\right)$ credits, we obtain:

$$\Delta R_{MC(k)}^*(\delta) \geq p \cdot \gamma(\Delta_{idle}^{k,min}) \tag{8.9}$$

Giving the maximum service curve property of γ and using Eq.(8.9), we have:

$$\begin{aligned}
\gamma(\delta) &\geq \Delta R_k^*(\delta) + \Delta R_{MC(k)}^*(\delta) \\
&\geq \Delta R_k^*(\delta) + p \cdot \gamma(\Delta_{idle}^{k,min})
\end{aligned}$$

Consequently, the upper bound of p is as follows:

$$p \leq \frac{\gamma(\delta) - \Delta R_k^*(\delta)}{\gamma(\Delta_{idle}^{k,min})} \tag{8.10}$$

When injecting Eq.(8.10) in Eq.(8.8), we obtain:

$$\Delta R_k^*(\delta) \leq \frac{\gamma(\delta) - \Delta R_k^*(\delta)}{\gamma(\Delta_{idle}^{k,min})} \cdot \gamma(\Delta_{send}^{k,max}) + \gamma(\Delta_{send,0}^{k,max})$$

We consider here a server with a constant rate C, thus:

$$\gamma(t) = C \cdot t$$

Hence, we obtain:

$$\Delta R_k^*(\delta) \cdot \left(1 + \frac{\Delta_{send}^{k,max}}{\Delta_{idle}^{k,min}}\right) \leq \frac{\delta}{\Delta_{idle}^{k,min}} \cdot C \cdot \Delta_{send}^{k,max} + C \cdot \Delta_{send,0}^{k,max}$$

$$\Delta R_k^*(\delta) \leq \frac{\frac{\delta}{\Delta_{idle}^{k,min}} \cdot C \cdot \Delta_{send}^{k,max} + C \cdot \Delta_{send,0}^{k,max}}{1 + \frac{\Delta_{send}^{k,max}}{\Delta_{idle}^{k,min}}}$$

$$\Delta R_k^*(\delta) \leq \frac{\Delta_{send}^{k,max}}{\Delta\gamma_k} \cdot C \cdot \delta + \Delta_{send,0}^{k,max} \cdot C \cdot \frac{\Delta_{idle}^{k,min}}{\Delta\gamma_k}$$

$$\text{where } \Delta\gamma_k = \Delta_{send}^{k,max} + \Delta_{idle}^{k,min}$$

8.4 Intuitive fluid models

In this section, our goal is to compute intuitive maximum and minimum service curves using a generalisation of the Achievable Worst-Cases. To keep the calculation manageable, we consider fluid (bit-per-bit) traffics and the 3-classes case study, where the SCT class is shaped by a BLS, presented in Section 3.4.1

8.4.1 Fluid minimum service Curve

In Chapter 4, we showed that the minimum service curve is not computed by considering all the traffic backlogged, but rather by inserting times when other traffics are not backlogged, as illustrated in Fig.8.1.

A generalisation of the ideas of Fig.8.5 is presented in Fig.8.6. It shows that intuitively, the most tight modelisation of the minimum service curve β_{SCT}^{bls}, offered by the BLS to SCT class, seems to be a linear function described in Fig.8.6 by a rate R and a latency T.

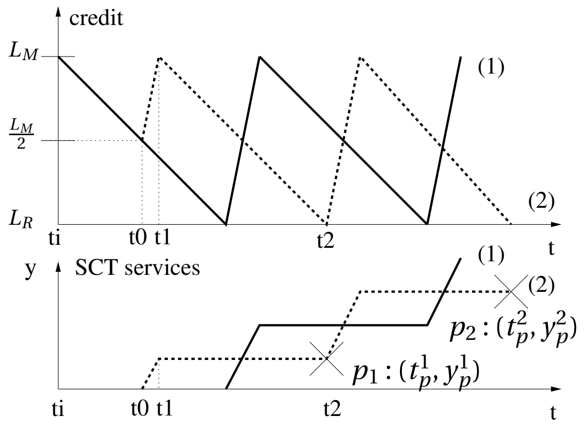


Figure 8.5: Two examples of worst-case BLS behaviours

In this section, we consider only fluid (bit-per-bit) traffic and our goal is to compute a intuitive maximum service curve $\beta_{SCT}^{bls,intui,fluid}$. The challenge is to prove we can find a scenario for each point of $\beta_{SCT}^{bls,intui,fluid} = R\cdot(t-T)^+$ as described in Fig.8.6. This will not be the proof that the $\beta_{SCT}^{bls,intui,fluid}$ described is the minimum service curve $\beta_{SCT}^{bls,fluid}$ but that it is an intuitive minimum service curve.

To verify the intuition proposed in Fig.8.6, we defined a family of curves inspired by line (2) in Fig.8.5, with the credit starting at $y = L_M$ and considering fluid traffic to simplify calculations. In particular, we compute the coordinates (t_p^n, y_p^n) of point p_n, with $n \in \mathbb{N}^*$, as described in Fig.8.5. It is the nth point where the credit reaches L_R. By varying $t0$ and $t1$ we wish to find points p_n forming the bold dotted linear line in Fig.8.6.

We defined the curve family as follows:

- We define δ the time between ti and $t0$. With $t0$ happening during the initial idle time: $\delta \leqslant \frac{L_M-L_R}{I_{idle}}$. The credit gained during δ is $\delta\cdot I_{idle}$;
- then the duration necessary for the credit to increase back to L_M at a rate $I_{send}$is $\delta\cdot\frac{I_{idle}}{I_{send}}$;
- then the duration necessary to decrease to L_R is $\frac{L_M-L_R}{I_{idle}}$;
- then the duration necessary to increase to L_M is $\frac{L_M-L_R}{I_{send}}$;
- the last two items are repeated indefinitely.

Finally, we have the nth point p_n:

$$t_p^n(\delta) = \delta + \delta\cdot\frac{I_{idle}}{I_{send}} + \frac{L_M-L_R}{I_{idle}} + (n-1)\cdot\left(\frac{L_M-L_R}{I_{idle}} + \frac{L_M-L_R}{I_{send}}\right)$$

Concerning y_p^n, the traffic transmission at a rate C happen between $t0$ and $t1$, and then during each duration necessary to go from L_R to L_M with a credit rate of I_{send}:

$$y_p^n(\delta) = \delta\cdot\frac{I_{idle}}{I_{send}}\cdot C + (n-1)\cdot\frac{L_M-L_R}{I_{send}}\cdot C$$

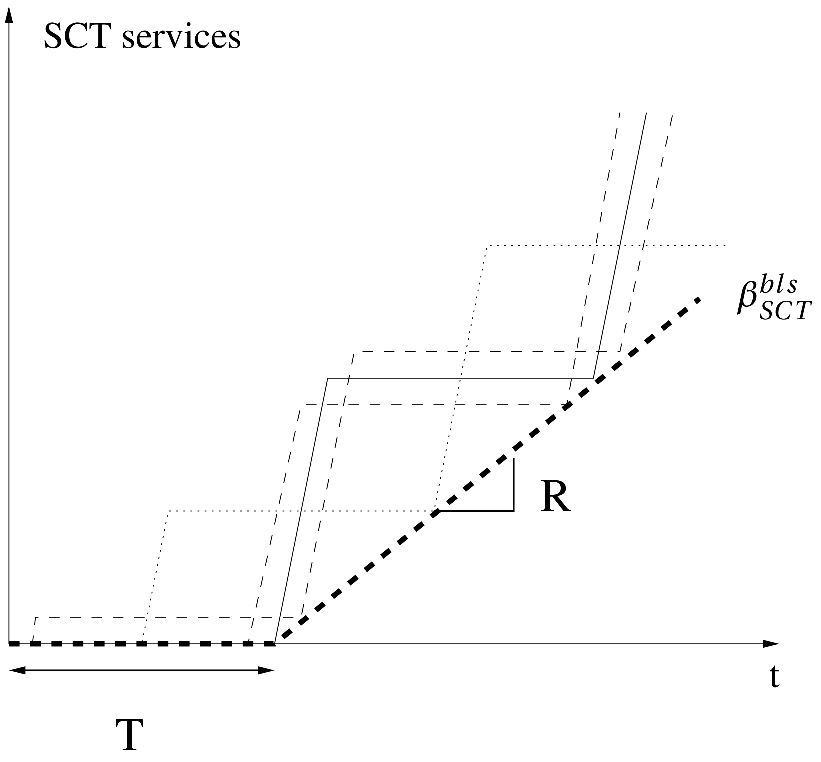


Figure 8.6: Minimum service curves: multiple BLS behaviour examples

First, we prove that when considering $\forall\delta$, $\forall n \in \mathbb{N}^*$, we obtain a continuous function $f(t)$. Then, we will find the expression of $f(t)$.

Let's consider a fixed n, then when δ varies, both $y_p^n(\delta)$ and $t_p^n(\delta)$ are continuous. Thus the resulting function $f_n(\delta)$ is continuous.

So, to obtain a continuous minimum service curve, we need to prove that when n varies the functions $f_n(\delta)$ form a continuous function $f(t)$. We do this by showing that $\forall n \in \mathbb{N}^*$, $\exists(\delta_1,\delta_2)$ such as $t_p^n(\delta_1) = t_p^{n+1}(\delta_2)$ and $y_p^n(\delta_1) = y_p^{n+1}(\delta_2)$.

As $f_n(\delta)$ are continuous, δ_1 and δ_2 are the extremities of the definition domain: 0 and $\frac{L_M-L_R}{I_{idle}}$. This gives:

$$\begin{aligned}
t_p^n(\frac{L_M-L_R}{I_{idle}}) &= \frac{L_M-L_R}{I_{idle}} + \frac{L_M-L_R}{I_{idle}}\cdot\frac{I_{idle}}{I_{send}} + \frac{L_M-L_R}{I_{idle}} \\
&\quad +(n-1)\cdot\left(\frac{L_M-L_R}{I_{idle}} + \frac{L_M-L_R}{I_{send}}\right) \\
&= \frac{L_M-L_R}{I_{idle}} + (n)\cdot\left(\frac{L_M-L_R}{I_{idle}} + \frac{L_M-L_R}{I_{send}}\right) \\
&= t_p^{n+1}(0)
\end{aligned}$$

and:

$$\begin{aligned} y_p^n(\frac{L_M - L_R}{I_{idle}}) &= \frac{L_M - L_R}{I_{idle}} \cdot \frac{I_{idle}}{I_{send}} \cdot C + (n-1) \cdot \frac{L_M - L_R}{I_{send}} \cdot C = (n) \cdot \frac{L_M - L_R}{I_{send}} \cdot C \\ &= y_p^{n+1}(0) \end{aligned}$$

We have now proved that the functions $f_n(t)$ form a continuous function $f(t)$ defined as, $\forall \delta$, $\forall n \in \mathbb{N}^*$:

$$f(t_p^n(\delta)) = y_p^n(\delta)$$

Using Fig.8.6, we suppose that $f(t)$ is probably a linear function. If we can find a and b such as $f(t) = a \cdot t + b$, then we have proved that the points p_n form a linear function when δ varies.

So, we search (a, b) such as:

$$\begin{aligned} y_p^n(\delta) &= t_p^n(\delta) \cdot a + b \\ \delta \cdot \frac{I_{idle}}{I_{send}} \cdot C + (n-1) \cdot \frac{L_M - L_R}{I_{send}} \cdot C &= \Bigg[\delta + \delta \cdot \frac{I_{idle}}{I_{send}} + \frac{L_M - L_R}{I_{idle}} \\ &\quad + (n-1) \cdot \left(\frac{L_M - L_R}{I_{idle}} + \frac{L_M - L_R}{I_{send}} \right) \Bigg] \cdot a + b \\ \delta \cdot \frac{I_{idle}}{I_{send}} \cdot C + (n-1) \cdot \frac{L_M - L_R}{I_{send}} \cdot C &= \delta \cdot a \cdot \left(1 + \frac{I_{idle}}{I_{send}} \right) + a \cdot \Bigg[\frac{L_M - L_R}{I_{idle}} \\ &\quad + (n-1) \cdot (L_M - L_R) \cdot \left(\frac{I_{idle} + I_{idle}}{I_{send} \cdot I_{idle}} \right) \Bigg] + b \end{aligned}$$

From this we deduce:

$$\begin{cases} \frac{I_{idle}}{I_{send}} \cdot C = a \cdot \left(1 + \frac{I_{idle}}{I_{send}} \right) \\ (n-1) \cdot \frac{L_M - L_R}{I_{send}} \cdot C = a \cdot \left[\frac{L_M - L_R}{I_{idle}} + (n-1) \cdot (L_M - L_R) \cdot \left(\frac{I_{idle} + I_{idle}}{I_{send} \cdot I_{idle}} \right) \right] + b \end{cases}$$

$$\begin{cases} \frac{I_{idle}}{I_{send}} \cdot C = a \cdot \left(\frac{I_{send} + I_{idle}}{I_{send}} \right) \\ b = (n-1) \cdot \frac{L_M - L_R}{I_{send}} \cdot C - a \cdot \left[\frac{L_M - L_R}{I_{idle}} + (n-1) \cdot (L_M - L_R) \cdot \left(\frac{C}{I_{send} \cdot I_{idle}} \right) \right] \end{cases}$$

$$\begin{cases} a = I_{idle} \\ b = -(L_M - L_R) \end{cases}$$

We have now defined $f(t)$ $\forall t$ such as $t \geqslant \frac{L_M - L_R}{I_{idle}}$ since $\delta \geqslant 0$ and $t_p^1(0) = \frac{L_M - L_R}{I_{idle}}$. Additionally, $\forall t \geqslant \frac{L_M - L_R}{I_{idle}}$, we know that $f(t)$ is always greater or equal to 0, thus $\forall t \geqslant 0$:

$$f(t) = I_{idle} \cdot \left(t - \frac{L_M - L_R}{I_{idle}} \right)^+ \tag{8.11}$$

Finally, we obtain the desired intuitive fluid minimum service curve such as:

$$\beta_{SCT}^{bls,intui,fluid}(t) = I_{idle} \cdot \left(t - \frac{L_M - L_R}{I_{idle}} \right)^+ \tag{8.12}$$

8.4.2 Fluid maximum service Curve

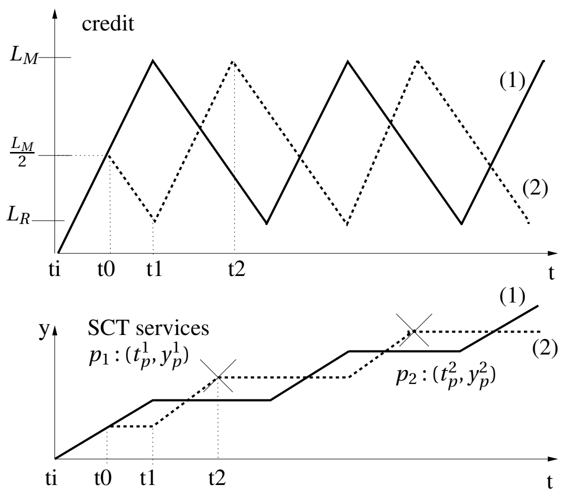


Figure 8.7: Two examples of best-case BLS behaviours

In Chapter 4, we showed that the maximum service curve is not computed by considering all the traffic backlogged, but rather by inserting times when other traffics are not backlogged, as illustrated in Fig.8.7.

A generalisation of the idea of Fig.8.7 is done in Fig.8.8. It shows that intuitively, the most tight modelisation of the maximum service γ_{SCT}^{bls}, offered by the BLS to SCT class when RC is backlogged, seems to be a piecewise linear function (bold dotted line described in Fig.8.8). The first piece is $C \cdot t$ during the time necessary for the credit to increase from 0 to L_M at a rate I_{send}: $t \leqslant \frac{L_M}{I_{send}}$, the second piece is a affine curve defined by a rate r and a burst b.

In this section, we consider only fluid (bit-per-bit) traffic and our goal is to compute a intuitive maximum service curve $\gamma_{SCT}^{bls,intui,fluid}$. The challenge is to prove we can find an intuitive scenario for each point of our $\gamma_{SCT}^{bls,intui,fluid}$ as described in Fig.8.8. This will not be the proof that $\gamma_{SCT}^{bls,intui,fluid}$ described here is the maximum fluid service but that it is an intuitive fluid maximum service curve.

To verify the intuition proposed in Fig.8.8, we defined a family of curves inspired by line (2) in Fig.8.7, with the credit starting at $y = 0$. In particular, we compute the coordinates (t_p^n, y_p^n) of point p_n, with $n \in \mathbb{N}^*$, as described in Fig.8.7. It is the nth point where the credit reaches L_M. According to our initial analysis, it seems to be part of the maximum service curve. By varying $t0$ and $t1$ we wish to find points p_n forming the bold dotted linear line in Fig.8.8.

We defined the curve family as follows:

- We define $\frac{L_R - 0}{I_{send}} + \delta$ the time between ti and $t0$. With $t0$ happening during the initial send time: $\delta \leqslant \frac{L_M - L_R}{I_{send}}$. The credit consumed during δ is $\delta \cdot I_{send}$;
- then the duration necessary for the credit to decrease back to L_R at a rate I_{idle} is $\delta \cdot \frac{I_{send}}{I_{idle}}$;

- then the duration necessary to increase to L_M is $\frac{L_M - L_R}{I_{send}}$;
- then the duration necessary to decrease to L_R is $\frac{L_M - L_R}{I_{idle}}$;
- the last two items are repeated indefinitely.

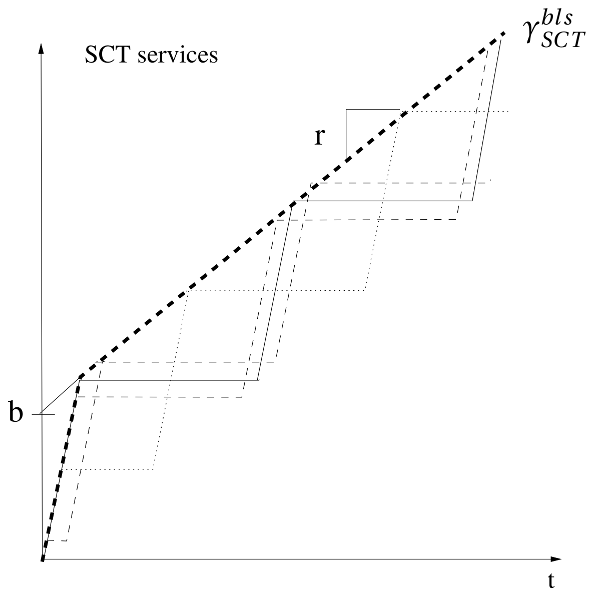


Figure 8.8: Maximum service curves when RC is backlogged: BLS behaviour examples

Finally, we have the nth point p_n defined as follows:

$$\begin{aligned} t_p^n(\delta) &= \frac{L_R - 0}{I_{send}} + \delta + \delta \cdot \frac{I_{send}}{I_{idle}} + \frac{L_M - L_R}{I_{send}} + (n-1) \cdot \left(\frac{L_M - L_R}{I_{idle}} + \frac{L_M - L_R}{I_{send}} \right) \\ &= \delta + \delta \cdot \frac{I_{send}}{I_{idle}} + \frac{L_M}{I_{send}} + (n-1) \cdot \left(\frac{L_M - L_R}{I_{idle}} + \frac{L_M - L_R}{I_{send}} \right) \end{aligned}$$

Concerning y_p^n, the traffic transmission times at a rate C happen between ti and $t0$, and then during each duration necessary to go from L_R to L_M with a credit rate of I_{send}:

$$\begin{aligned} y_p^n(\delta) &= \frac{L_R - 0}{I_{send}} \cdot C + \delta \cdot C + \frac{L_M - L_R}{I_{send}} \cdot C + (n-1) \cdot \frac{L_M - L_R}{I_{send}} \cdot C \\ &= \delta \cdot C + \frac{L_M}{I_{send}} \cdot C + (n-1) \cdot \frac{L_M - L_R}{I_{send}} \cdot C \end{aligned}$$

First, we prove that when considering $\forall\ \delta$ such as $\frac{L_M - L_R}{I_{send}} \geqslant \delta \geqslant 0$, and $\forall\ n \in \mathbb{N}^*$, we obtain a continuous function $f(t)$. Then, we will find the expression of $f(t)$.

As before, let's consider a fixed n, then when δ varies, both $y_p^n(\delta)$ and $t_p^n(\delta)$ are continuous. Thus the resulting function $f_n(\delta)$ is continuous.

Again, to obtain a continuous minimum service curve, we need to prove that when n varies the functions $f_n(\delta)$ form a continuous function $f(t)$. We do this by showing that $\forall n \in \mathbb{N}^*$, $\exists(\delta_1,\delta_2)$ such as $t_p^n(\delta_1) = t_p^{n+1}(\delta_2)$ and $y_p^n(\delta_1) = y_p^{n+1}(\delta_2)$.

As $f_n(\delta)$ are continuous, δ_1 and δ_2 are the extremities of the definition domain: 0 and $\frac{L_M - L_R}{I_{send}}$. This gives:

$$\begin{aligned}
t_p^n(\frac{L_M - L_R}{I_{send}}) &= \frac{L_M - L_R}{I_{send}} + \frac{L_M - L_R}{I_{send}} \cdot \frac{I_{send}}{I_{idle}} + \frac{L_M}{I_{send}} \\
&\quad +(n-1)\cdot\left(\frac{L_M - L_R}{I_{idle}} + \frac{L_M - L_R}{I_{send}}\right) \\
&= \frac{L_M}{I_{send}} + (n)\cdot\left(\frac{L_M - L_R}{I_{idle}} + \frac{L_M - L_R}{I_{send}}\right) \\
&= t_p^{n+1}(0)
\end{aligned}$$

and:

$$\begin{aligned}
y_p^n(\frac{L_M - L_R}{I_{send}}) &= \frac{L_M - L_R}{I_{send}} \cdot C + \frac{L_M}{I_{send}} \cdot C + (n-1)\cdot \frac{L_M - L_R}{I_{send}} \cdot C \\
&= \frac{L_M}{I_{send}} \cdot C + (n) \cdot \frac{L_M - L_R}{I_{send}} \cdot C \\
&= y_p^{n+1}(0)
\end{aligned}$$

We have now proven that the functions $f_n(t)$ form a continuous function $f(t)$ defined as, $\forall\delta$, $\forall n \in \mathbb{N}^*$:

$$f(t_p^n(\delta)) = y_p^n(\delta)$$

From Fig.8.8, we suppose that $f(t)$ is a affine function $\forall\ t \geqslant \frac{L_M}{I_{send}}$. If we can find a and b such as $f(t) = a \cdot t + b$, then we have proved that the points p_n form a linear curve when δ varies.

So, we search (a,b) such as:

$$\begin{aligned}
y_p^n(\delta) &= t_p^n(\delta)\cdot a + b \qquad (8.13) \\
\delta \cdot C + \frac{L_M}{I_{send}} \cdot C + (n-1) \cdot \frac{L_M - L_R}{I_{send}} \cdot C &= \left[\delta + \delta \cdot \frac{I_{send}}{I_{idle}} + \frac{L_M}{I_{send}}\right. \\
&\quad \left. +(n-1)\cdot\left(\frac{L_M - L_R}{I_{idle}} + \frac{L_M - L_R}{I_{send}}\right)\right] \cdot a + b \\
\delta \cdot C + \frac{L_M}{I_{send}} \cdot C + (n-1) \cdot \frac{L_M - L_R}{I_{send}} \cdot C &= \delta \cdot a \cdot \left(1 + \frac{I_{send}}{I_{idle}}\right) + a \cdot \left[\frac{L_M}{I_{send}}\right. \\
&\quad \left. +(n-1)\cdot(L_M - L_R)\cdot\left(\frac{I_{idle} + I_{idle}}{I_{send}\cdot I_{idle}}\right)\right] + b
\end{aligned}$$

From this we deduce:

$$\begin{cases} C = a \cdot \left(1 + \frac{I_{send}}{I_{idle}}\right) \\ \frac{L_M}{I_{send}} \cdot C + (n-1) \cdot \frac{L_M - L_R}{I_{send}} \cdot C = a \cdot \left[\frac{L_M}{I_{send}} + (n-1)\cdot(L_M - L_R)\cdot\left(\frac{I_{idle}+I_{idle}}{I_{send}\cdot I_{idle}}\right)\right] + b \end{cases}$$

$$\begin{cases} C = a \cdot \left(\frac{I_{send}+I_{idle}}{I_{idle}}\right) \\ b = \frac{L_M}{I_{send}} \cdot C + (n-1) \cdot \frac{L_M - L_R}{I_{send}} \cdot C - a \cdot \left[\frac{L_M}{I_{send}} + (n-1) \cdot (L_M - L_R) \cdot \left(\frac{I_{idle}+I_{idle}}{I_{send} \cdot I_{idle}}\right)\right] \end{cases}$$

$$\begin{cases} a = I_{idle} \\ b = \frac{L_M}{I_{send}} \cdot C - I_{idle} \cdot \frac{L_M}{I_{send}} = L_M \end{cases}$$

We have now defined $f(t)$ $\forall t$ such as $t \geqslant \frac{L_M - L_R}{I_{send}}$ since $\delta \geqslant 0$ and $t_p^1(0) = \frac{L_M - L_R}{I_{send}}$.

Additionally, $\forall t \leqslant \frac{L_M - L_R}{I_{send}}$, we know that $f(t) = C \cdot t$ thus we define f such as:

$$f(t) = \begin{cases} \forall t \geqslant \frac{L_M - L_R}{I_{send}} : I_{idle} \cdot t + L_M \\ otherwise\colon C \cdot t \end{cases}$$

As $\forall$ $t \leqslant \frac{L_M - L_R}{I_{send}}$: $I_{idle} \cdot t + L_M \geqslant C \cdot t$, this gives: $f(t) = \min(C \cdot t, I_{idle} \cdot t + L_M)$.

It is worth noting that when computing the minimum RC service curve, according to Theorem 7, if f(t) is a maximum service curve we have:

$$\begin{aligned} \beta_{RC}(t) &= \left(C \cdot t - \min(\min(C \cdot t, I_{idle} \cdot t + L_M), \alpha_{SCT} \oslash \beta_{SCT}^{bls}(t)) - \max_{i \in RC \cup BE} MFS_i\right)_{\uparrow} \\ &= \left(C \cdot t - \min(C \cdot t, I_{idle} \cdot t + L_M, \alpha_{SCT} \oslash \beta_{SCT}^{bls}(t)) - \max_{i \in RC \cup BE} MFS_i\right)_{\uparrow} \\ &= \Big(\max(C \cdot t - C \cdot t, C \cdot t - I_{idle} \cdot t - L_M, C \cdot t - \alpha_{SCT} \oslash \beta_{SCT}^{bls}(t)) \\ &\quad - \max_{i \in RC \cup BE} MFS_i\Big)_{\uparrow} \\ &= \Big(\max(-\max_{i \in RC \cup BE} MFS_i, C \cdot t - I_{idle} \cdot t - L_M - \max_{i \in RC \cup BE} MFS_i, \\ &\quad C \cdot t - \alpha_{SCT} \oslash \beta_{SCT}^{bls}(t) - \max_{i \in RC \cup BE} MFS_i)\Big)_{\uparrow} \end{aligned}$$

As $-\max_{i \in RC \cup BE} MFS_i < 0$ and $\beta_{RC}(t) \geqslant 0$ this gives:

$$\beta_{RC}(t) = \left(\max(C \cdot t - I_{idle} \cdot t - L_M, C \cdot t - \alpha_{SCT} \oslash \beta_{SCT}^{bls}) - \max_{i \in RC \cup BE} MFS_i\right)_{\uparrow}$$

Hence, if $f(t) = \min(C \cdot t, I_{idle} \cdot t + L_M)$ is a maximum service curve for SCT traffic in the BLS node, then $f_{linear}(t) = I_{idle} \cdot t + L_M$ is a maximum service curve that gives the same minimum RC service curve as the piecewise version, $f(t)$.

Finally, we obtain the desired intuitive fluid maximum service curve such as:

$$\gamma_{SCT}^{bls,intui,fluid}(t) = I_{idle} \cdot t + L_M \tag{8.14}$$

8.5 Continuous-Credit-based Approach (CCbA) model proofs

In this section, we detail the proofs of the strict minimum and maximum service curves. Both proofs are based on three lemmas presented in the next section.

8.5.1 Continuous-credit Lemmas

We denote $R_k^*(t)$ the output cumulative function of the class k traffic, and $\Delta R_k^*(\delta)$ its variation during an interval δ.

The BLS credit tries to keep an accurate accounting of the traffic sent. There are two situations when it loses track due to non-preempted transmissions:

1. when the credit reaches L_M^k and the current class k frame has not finished its transmission;

2. when the credit reaches 0 and the current frame is still being transmitted.

We call this the saturation of the credit, either at L_M^k by class k traffic, or at 0 by other traffics. The saturation at L_M^k can only occur when a class k frame is being transmitted, while the saturation at 0 can not occur when a class k frame is being transmitted.

Hence, we call $\Delta R^*_{L_M^k,sat}(\delta)$ (resp.$\Delta R^*_{0,sat}(\delta)$) the part of $\Delta R_k^*(\delta)$ (resp. $\delta \cdot C - \Delta R_k^*(\delta)$), that can be sent during any interval δ while the credit is saturated at L_M^k (resp. at 0).

We present here three lemmas linked to the credit saturation and necessary to the service curve proofs. First in Lemma 1, we show how to bound the sum of the credit consumed and the credit gained, depending on the credit saturations. Then, we detail the bounds of the credit saturations at L_M^k in Lemma 2, and at 0 in Lemma 3.

Lemma 1 (Continuous credit bounds)**.** *We consider a shaped class k, with a maximum credit level L_M^k. $\forall\delta$, computing the sum of the credit consumed and gained give the following inequations:*

$$L_M^k \geqslant \begin{pmatrix} \Delta R_k^*(\delta) - \dfrac{\Delta R^*_{L_M^k,sat}(\delta)}{C} \cdot I_{send}^k \\ -(\delta - \dfrac{\Delta R^*_{0,sat}(\delta)}{C}) \cdot I_{idle}^k \end{pmatrix} \geqslant -L_M^k$$

Proof. In an interval δ, the accurate consumed credit is the time it takes to send the non-saturating traffic $\frac{\Delta R_k^*(\delta) - \Delta R^*_{L_M^k,sat}(\delta)}{C}$ multiplied by the sending slope:

$$credit_{consumed}^k = \left(\frac{\Delta R_k^*(\delta) - \Delta R^*_{L_M^k,sat}(\delta)}{C} \right) \cdot I_{send}^k$$

And conversely, the accurate gained credit is the remaining time $\delta - \frac{\Delta R_k^*(\delta)}{C}$ minus the saturation time $\frac{\Delta R_{0,sat}^*(\delta)}{C}$, multiplied by the signed idle slope:

$$credit_{gained}^k = \left(\delta - \frac{\Delta R_k^*(\delta) + \Delta R_{0,sat}^*(\delta)}{C}\right) \cdot (-I_{idle}^k)$$

Thus $\forall \delta \in \mathbb{R}^+$, using the fact that $I_{send}^k + I_{idle}^k = C$, the sum of the gained credit and the consumed credit is:

$$\begin{aligned} credit_{consumed}^k + credit_{gained}^k &= \left(\frac{\Delta R_k^*(\delta) - \Delta R_{L_M^k,sat}^*(\delta)}{C}\right) \cdot (I_{send}^k) \\ &\quad + \left(\delta - \frac{\Delta R_k^*(\delta) + \Delta R_{0,sat}^*(\delta)}{C}\right) \cdot (-I_{idle}^k) \\ &= \Delta R_k^*(\delta) - \frac{\Delta R_{L_M^k,sat}^*(\delta)}{C} \cdot I_{send}^k \\ &\quad - \left(\delta - \frac{\Delta R_{0,sat}^*(\delta)}{C}\right) \cdot I_{idle}^k \end{aligned}$$

We know that the credit is a continuous function with a lower bound: 0 and an upper bound L_M^k. So the sum of credit consumed and gained is always bounded by $-L_M^k$ and $+L_M^k$.

$$\begin{aligned} L_M^k \geqslant \quad & credit_{consumed}^k + credit_{gained}^k && \geqslant -L_M^k \\ L_M^k \geqslant \quad & \begin{pmatrix} \Delta R_k^*(\delta) - \frac{\Delta R_{L_M^k,sat}^*(\delta)}{C} \cdot I_{send}^k \\ -\left(\delta - \frac{\Delta R_{0,sat}^*(\delta)}{C}\right) \cdot I_{idle}^k \end{pmatrix} && \geqslant -L_M^k \end{aligned}$$

□

Lemma 2 (credit saturation at 0). *We consider a shaped class k, with the aggregate traffic of priority strictly higher than $p_H(k)$ α_h-constrained with $\alpha_h = r_h \cdot t + b_h$.*

$\forall \delta$, the amount of traffic sent while the traffic is saturated at 0 is such as:

$$0 \leqslant \Delta R_{0,sat}^*(\delta) \leqslant \begin{pmatrix} \sum_{h \in HC(k)} r_h \cdot \delta + b_h \\ + MFS_{MC(k)}^{sat} \cdot \left(\frac{\delta}{\Delta_{inter}^{k,\beta}} + 1\right) \end{pmatrix}$$

with:

$$MFS_{MC(k)}^{sat} = \max\left(\max_{j \in MC(k)} MFS_j - \frac{C}{I_{idle}^k} \cdot L_R^k, 0\right)$$

$$\Delta_{inter}^{k,\beta} = \frac{\max_{j \in MC(k)} MFS_j}{C} + \frac{L_M^k - L_R^{k,min}}{I_{send}^k} + \frac{L_M^k - L_R^k}{I_{idle}^k}$$
$$L_R^{k,min} = \max(L_R^k - \frac{\max_{j \in MC(k)} MFS_j}{C} \cdot I_{idle}^k, 0)$$

Proof. First, we know that $\Delta R_{0,sat}^*(\delta) \geqslant 0$. Secondly, we consider the impact of MC(k) and HC(k), the impact of LC(k) being taken into account in $\beta_k^{sp}(t)$ to compute an upper bound.

Impact of MC(k) on $\Delta R_{0,sat}^*(\delta)$

In the presence of class k frames, the saturation of the credit at 0 can occur if an additional frame is sent while the credit is decreasing and about to reach L_R^k. Due to non-preemption, the frame finishes its transmission even though the class k priority is now higher.

To be able to compute the largest impact of the non-preemption of MC(k) frames on class k traffic, we must find the highest number of non-preempted frames that can be sent during a time interval δ. Then, we must compute the part of the non-preempted frame sent while the credit is saturated.

So first, we must compute the smallest duration between two occurrences of the phenomenon. Fig.8.9 illustrates the following explanation.

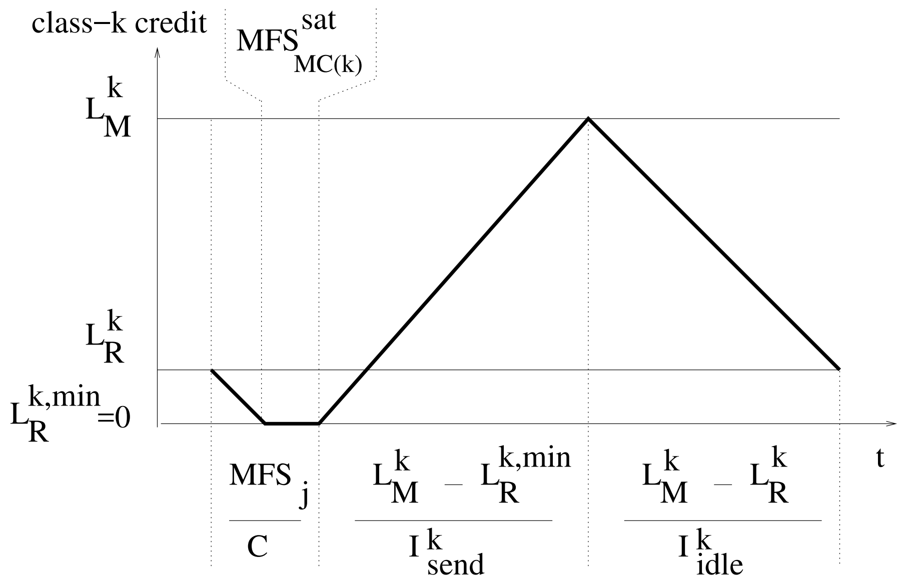


Figure 8.9: Computing $\beta_k^{bls}(t)$

After the first non-preempted MC(k) frame has been sent, the priority of the class k queue is high. So, in presence of class k traffic, no MC(k) traffic can be sent until a priority change: L_M^k must be reached between two non-preempted MC(k) frames.

Thus, we study the intervals of time between the start of two transmissions of non-preempted MC(k) frames starting their transmission just before L_R^k is reached. The smallest duration of such an interval is equal to the sum of

1. the transmission time of the non-preempted MC(k) frame , such as at the end of the transmission the credit reaches $L_R^{k,min} = \max(L_R^k - \frac{\max_{j\in MC(k)} MFS_j}{C} \cdot I_{idle}^k, 0)$;
2. the duration $\frac{L_M^k - L_R^{k,min}}{I_{send}^k}$ because class-k traffic has to be sent continuously in order for the credit to reach L_M^k in the minimum duration ;
3. finally $\frac{L_M^k - L_R^k}{I_{idle}^k}$ because MC(k) traffic has to be sent continuously in order for the credit to return in the minimum duration to L_R^k.

In total, the minimum duration between the start of the transmission of two non-preempted MC(k) frames (each starting just before L_R^k is reached), is

$$\Delta_{inter}^{k,\beta} = \frac{\max_{j\in MC(k)} MFS_j}{C} + \frac{L_M^k - L_R^{k,min}}{I_{send}^k} + \frac{L_M^k - L_R^k}{I_{idle}^k}$$

Thus during δ, the number of time a non-preempted MC(k) frame can be sent is upper bounded by $\lceil \frac{\delta}{\Delta_{inter}^{k,\beta}} \rceil$.

Secondly, we need to compute the maximum amount data sent while the credit remains at 0 during the transmission of one non-preempted maximum-sized MC(k) frame as illustrated in Fig.8.9. This is equal to the maximum size of a MC(k) frame, minus the amount of data transmitted while the credit decreases from L_R^k to 0:

$$MFS_{MC(k)}^{sat} = \max(\max_{j\in MC(k)} MFS_j - \frac{C}{I_{idle}^k} \cdot L_R^k, 0)$$

Impact of HC(k) on $\Delta R_{0,sat}^*(\delta)$

The second way credit can saturate at 0 happens if traffic from HC(k) is sent while the credit remains at 0. We denoted $\alpha_h(t)$ the aggregate traffic of HC(k), arriving at a rate of r_h, with a burst b_h, such as $\alpha_h(t) = r_h \cdot \delta + b_h$.

As a result, the amount of MC(k) and HC(k) traffic sent while the credit is saturated is such as:

$$\begin{aligned}\Delta R_{0,sat}^*(\delta) &\leqslant \sum_{h\in HC(k)} r_h \cdot \delta + b_h + MFS_{MC(k)}^{sat} \cdot \lceil \frac{\delta}{\Delta_{inter}^{k,\beta}} \rceil \\ &\leqslant \sum_{h\in HC(k)} r_h \cdot \delta + b_h + MFS_{MC(k)}^{sat} \cdot \left(\frac{\delta}{\Delta_{inter}^{k,\beta}} + 1 \right)\end{aligned}$$

□

Lemma 3 (credit saturation at L_M^k)**.** *We consider a shaped class k, with the aggregate traffic of priority strictly higher than $p_H(k)$ α_h-constrained with $\alpha_h = r_h \cdot t + b_h$.*

$\forall \delta$, the amount of traffic sent while the traffic is saturated at L_M^k is such as:

$$0 \leqslant \Delta R^*_{L_M^k,sat}(\delta) \leqslant MFS_k \cdot \left(\frac{\delta}{\Delta^{k,\gamma}_{inter}} + 1 \right)$$

with:

$$\Delta^{k,\gamma}_{inter} = \frac{MFS_k}{C} + \frac{L_M^k - L_R^k}{I^k_{idle}} + \frac{L_M^k - L_R^k}{I^k_{send}}$$

Proof. First, we know that $\Delta R^*_{L_M^k,sat}(\delta) \geqslant 0$. Secondly for the upper bound, in the presence of MC(k) frames, the saturation of the credit at L_M^k can only occur if an additional frame is sent while the credit is increasing and about to reach L_M^k. Due to non-preemption, the frame finishes its transmission even though the class k priority is now lower.

To be able to compute the largest impact of the non-preemption of class k frames, we must find the highest number of non-preempted frames that can be sent during a time interval δ. So we must compute the smallest duration between two occurrences of the phenomenon. Fig.8.10 illustrates the following explanation. After the first non-preempted class-k frame has been sent, the priority of the class k queue is low, so in presence of MC(k) traffic, no class k traffic can be sent until a priority change: L_R^k must be reached between two non-preempted class-k frames. Thus, we study the intervals of time between the start of two transmissions of non-preempted class-k frames starting their transmission just before L_M^k is reached. The smallest duration of such an interval is equal to the sum of:

1. the transmission time of the non-preempted class-k frame (at the end of the transmission the credit is equal to L_M^k);
2. the duration $\frac{L_M^k - L_R^k}{I^k_{idle}}$ because MC(k) traffic has to be sent continuously in order for the credit to reach L_R^k in the minimum duration ;
3. finally $\frac{L_M^k - L_R^k}{I^k_{send}}$ because class k traffic has to be sent continuously in order for the credit to return in the minimum duration to L_M^k.

In total, the minimum duration between the start of the transmission of two non-preempted class-k frames (each starting just before L_M^k is reached), is

$$\Delta^{k,\gamma}_{inter} = \frac{MFS_k}{C} + \frac{L_M^k - L_R^k}{I^k_{idle}} + \frac{L_M^k - L_R^k}{I^k_{send}}$$

Thus during δ, the number of time a non-preempted class-k frame can be sent is upper bounded by $\lceil \frac{\delta}{\Delta^{k,\gamma}_{inter}} \rceil$.

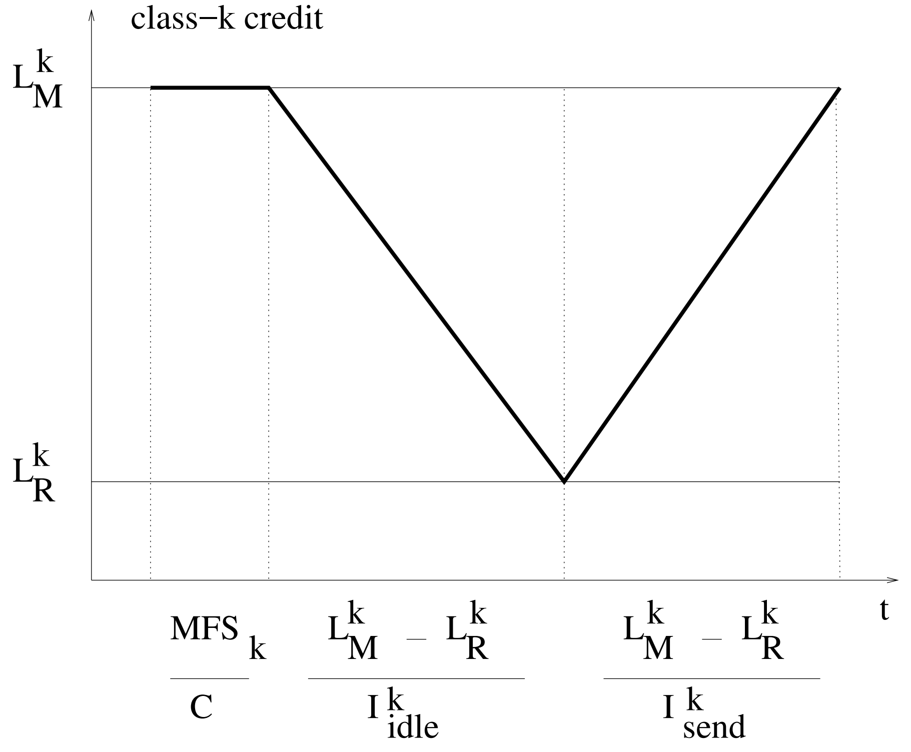

class−k credit
L^k_M
L^k_R
t
$\frac{MFS_k}{C}$
$\frac{L^k_M - L^k_R}{I^k_{idle}}$
$\frac{L^k_M - L^k_R}{I^k_{send}}$

Figure 8.10: Computing $\gamma_k^{bls}(t)$

As a result, the amount class k traffic sent while the credit is saturated is such as:

$$\Delta R^*_{L^k_M,sat}(\delta) \leqslant MFS_k \cdot \lceil \frac{\delta}{\Delta^{k,\gamma}_{inter}} \rceil \leqslant MFS_k \cdot \left(\frac{\delta}{\Delta^{k,\gamma}_{inter}} + 1 \right)$$

□

8.5.2 Th.8: CCbA strict minimum service curve

We search a strict minimum service curve offered to a class k defined by a Rate-Latency curve, i.e., $\beta_k^{bls}(t) = \rho \cdot (t - \tau)^+$ with rate ρ and initial latency τ.

The impact of other classes, are separated into three parts: the impact of $LC(k)$, $MC(k)$, $HC(k)$.

According to the definition of the strict minimum service curve, $\forall$ backlogged period δ:

$$\Delta R^*_k(\delta) \geqslant \beta_k^{bls}(\delta) = \rho \cdot (\delta - \tau)^+ \tag{8.15}$$

For any duration lower than τ, the variation of the output is lower bounded by 0.

$$\forall \delta \leqslant \tau, \Delta R^*_k(\delta) \geqslant 0$$

Thus, the best τ for our strict service curve is the largest duration during which no class k traffic can be sent. So, when considering the impacts of the different classes we have:

1. for traffic of Lower Classes $LC(k)$, the impact is the one computed with Static Priority: it is due to the non-preemption and is taken into account in the Static Priority model;
2. for traffic of Medium Classes $MC(k)$: the worst-case occurs if the credit starts at L_M^k, MC(k) frames are transmitted until L_R^k is reached and due to non-preemption an additional MC(f) frame is sent. We denote this duration $\Delta_{idle}^{k,\beta}$. So, we have:
$$\Delta_{idle}^{k,\beta} = \frac{L_M^k - L_R^k}{I_{idle}^k} + \frac{\max_{j\in MC(k)} MFS_j}{C}$$
3. for Higher Classes $HC(k)$, the impact is already computed with Static Priority and is not taken into account here.

So finally, we have:
$$\tau = \Delta_{idle}^{k,\beta}$$

Concerning the ρ, we search for a strictly positive rate. We use the definition of β_k^{bls} as a Rate-Latency strict service curve and Eq.(8.15) to deduce a property of ρ. We notice the limit toward infinity of $\Delta R_k^*(\delta)$ over δ will be greater than ρ:
$$\lim_{\delta\to+\infty} \frac{\Delta R_k^*(\delta)}{\delta} \geqslant \lim_{\delta\to+\infty} \rho\cdot\left(1-\frac{\tau}{\delta}\right) = \rho.$$

So we look for a $x > 0$ fulfilling the following condition:
$$\lim_{\delta\to+\infty} \frac{\Delta R_k^*}{\delta} \geqslant x.$$

We now use the continuity property of the BLS credit to determine x. From Lemma 1, we know that:
$$\begin{pmatrix} \Delta R_k^*(\delta) - \dfrac{\Delta R_{L_M^k,sat}^*(\delta)}{C}\cdot I_{send}^k \\ -(\delta - \dfrac{\Delta R_{0,sat}^*(\delta)}{C})\cdot I_{idle}^k \end{pmatrix} \geqslant -L_M^k$$

Thus:
$$\Delta R_k^*(\delta) \geqslant -L_M^k + \frac{\Delta R_{L_M^k,sat}^*(\delta)}{C}\cdot I_{send}^k + (\delta - \frac{\Delta R_{0,sat}^*(\delta)}{C})\cdot I_{idle}^k$$

To find x, we must find a lower bound of $\lim\limits_{\delta\to+\infty} \frac{\Delta R_k^*(\delta)}{\delta}$, so we have:
$$\frac{\Delta R_k^*(\delta)}{\delta} \geqslant \frac{-L_M^k}{\delta} + \frac{\Delta R_{L_M^k,sat}^*(\delta)}{\delta\cdot C}\cdot I_{send}^k + (1 - \frac{\Delta R_{0,sat}^*(\delta)}{\delta\cdot C})\cdot I_{idle}^k$$

$$\lim_{\delta\to+\infty} \frac{\Delta R_k^*(\delta)}{\delta} \geqslant \lim_{\delta\to+\infty} \frac{-L_M^k}{\delta} + \frac{\Delta R^*_{L_M^k,sat}(\delta)}{\delta\cdot C}\cdot I^k_{send} + (1-\frac{\Delta R^*_{0,sat}(\delta)}{\delta\cdot C})\cdot I^k_{idle} \tag{8.16}$$

We need the lower bound of $\Delta R^*_{L_M^k,sat}(\delta)$, and the upper bound of $\Delta R^*_{0,sat}(\delta)$. We use Lemmas 2 and 3 to compute the bounds. This gives:

$$\lim_{\delta\to\infty} \frac{\Delta R^{*,max}_{L_M^k,sat}(\delta)}{\delta} \geqslant 0 \tag{8.17}$$

$$\lim_{\delta\to\infty} \frac{\Delta R^{*,max}_{0,sat}(\delta)}{\delta} \leqslant \sum_{h\in HC(k)} r_h + \frac{MFS^{sat}_{MC(k)}}{\Delta^{k,\beta}_{inter}} \tag{8.18}$$

Thus, from Eq.(8.16), Eq.(8.17), and Eq.(8.18), we deduce:

$$\lim_{\delta\to+\infty} \frac{\Delta R_k^*(\delta)}{\delta} \geqslant \lim_{\delta\to+\infty} (1-\frac{\Delta R^{*,max}_{0,sat}(\delta)}{\delta\cdot C})\cdot I^k_{idle} = \left(C - \sum_{h\in HC(k)} r_h - \frac{MFS^{sat}_{MC(k)}}{\Delta^{k,\beta}_{inter}}\right)\cdot\frac{I^k_{idle}}{C}$$

Finally, we have found a suitable ρ such as: $\lim_{\delta\to+\infty} \frac{\Delta R_k^*(\delta)}{\delta} \geqslant \rho$ with

$$\rho = \left(C - \sum_{h\in HC(k)} r_h - \frac{MFS^{sat}_{MC(k)}}{\Delta^{k,\beta}_{inter}}\right)\cdot\frac{I^k_{idle}}{C}.$$

8.5.3 Th.9: CCbA maximum service curve

We search a maximum service curve offered to a class k defined by a leaky-bucket curve, i.e., $\gamma_k^{bls}(t) = r\cdot t + b$ with rate r and burst b.

From [87], we know that $\Delta R_k^*(t-s) \leqslant B(s) + \gamma(t-s)$, with $B(s)$ the backlog at s. We search for z such as $\Delta R_k^*(t-s) \leqslant z$. As $B(s) \geqslant 0$, we obtain:

$$\Delta R_k^*(t-s) \leqslant z \leqslant B(s) + z$$

Hence, with $\delta = t-s$, we select $\gamma(\delta) = z$, which gives:

$$\Delta R_k^*(\delta) \leqslant \gamma_k^{bls}(\delta) \tag{8.19}$$

In the absence of other traffic, class k can use the full capacity of the link, so $\Delta R_k^*(\delta) \leqslant C\cdot t$. Thus, we deduce that:

$$\gamma_k^{bls}(t) = C\cdot t.$$

In a MC(k) backlogged period, we use the definition of γ_k^{bls} as a leaky-bucket maximum service curve to deduce a property of r using Eq.(8.19).

Computing r
We notice the limit toward infinity of ΔR_k^* over δ will be lower than r:

$$\lim_{\delta\to+\infty} \frac{\Delta R_k^*}{\delta} \leqslant \lim_{\delta\to+\infty} r + \frac{b}{\delta} = r$$

So we search for a strictly positive rate x, equal or lower than the link output rate C fulfilling the following condition:

$$\lim_{\delta\to+\infty} \frac{\Delta R_k^*}{\delta} \leqslant x$$

We use the continuity property of the BLS credit to determine x. From Lemma 1, we know that:

$$\Delta R_k^*(\delta) - \frac{\Delta R^*_{L_M^k,sat}(\delta)}{C} \cdot I^k_{send} - (\delta - \frac{\Delta R^*_{0,sat}(\delta)}{C}) \cdot I^k_{idle} \leqslant L_M^k$$

Thus,

$$\Delta R_k^*(\delta) \leqslant L_M^k + \frac{\Delta R^*_{L_M^k,sat}(\delta)}{C} \cdot I^k_{send} + (\delta - \frac{\Delta R^*_{0,sat}(\delta)}{C}) \cdot I^k_{idle}$$

To find x, we must find a lower bound of $\lim\limits_{\delta\to+\infty} \frac{\Delta R_k^*(\delta)}{\delta}$, so we have:

$$\frac{\Delta R_k^*(\delta)}{\delta} \leqslant \frac{L_M^k}{\delta} + \frac{\Delta R^*_{L_M^k,sat}(\delta)}{\delta \cdot C} \cdot I^k_{send} + (1 - \frac{\Delta R^*_{0,sat}(\delta)}{\delta \cdot C}) \cdot I^k_{idle}$$

$$\lim_{\delta\to+\infty} \frac{\Delta R_k^*(\delta)}{\delta} \leqslant \lim_{\delta\to+\infty} \frac{L_M^k}{\delta} + \frac{\Delta R^*_{L_M^k,sat}(\delta)}{\delta \cdot C} \cdot I^k_{send} + (1 - \frac{\Delta R^*_{0,sat}(\delta)}{\delta \cdot C}) \cdot I^k_{idle}$$

We need the lower bound of $\Delta R^*_{0,sat}(\delta)$, and the upper bound of $\Delta R^*_{L_M^k,sat}(\delta)$. We use Lemmas 2 and 3 to compute the bounds. This gives:

$$\lim_{\delta\to\infty} \frac{\Delta R^{*,max}_{L_M^k,sat}(\delta)}{\delta} \geqslant 0$$

$$\lim_{\delta\to\infty} \frac{\Delta R^{*,max}_{L_M^k,sat}(\delta)}{\delta} \leqslant \frac{MFS_k}{\Delta^{k,\gamma}_{inter}}$$

Thus, from Eq.(8.20), we deduce:

$$\lim_{\delta\to+\infty} \frac{\Delta R_k^*(\delta)}{\delta} \leqslant \lim_{\delta\to+\infty} I^k_{idle} + \frac{\Delta R^{*,max}_{L_M^k,sat}(\delta)}{\delta \cdot C} \cdot I^k_{send} = I^k_{idle} + \frac{MFS_k}{\Delta^{k,\gamma}_{inter}} \cdot \frac{I^k_{send}}{C}$$

We call $\Delta_{inter}^{k,\gamma,send}$ the interval during which class k frames are sent, and $\Delta_{inter}^{k,\gamma,idle}$ the interval during which MC(k) frames are sent such as $\Delta_{inter}^{k,\gamma} = \Delta_{send}^{k,\gamma} + \Delta_{idle}^{k,\gamma}$.

$$\Delta_{send}^{k,\gamma} = \frac{MFS_k}{C} + \frac{L_M^k - L_R^k}{I_{send}^k}$$

$$\Delta_{idle}^{k,\gamma} = \frac{L_M^k - L_R^k}{I_{idle}^k}$$

Using the definitions of the different expressions, we deduce that:

$$I_{idle}^k + \frac{MFS_k}{\Delta_{inter}^{k,\gamma}} \cdot \frac{I_{send}^k}{C} = \frac{\Delta_{send}^{k,\gamma}}{\Delta_{inter}^{k,\gamma}} \cdot C < C$$

Finally, we have found a suitable r, such as: $\lim\limits_{\delta \to +\infty} \frac{\Delta R_k^*(\delta)}{\delta} \leqslant r$ with $r = \frac{\Delta_{send}^{k,\gamma}}{\Delta_{inter}^{k,\gamma}} \cdot C$

Now that we have found r, we need to find b such as $\forall$ MC(k) backlogged period δ:

$$\Delta R_k^*(\delta) \leqslant \frac{\Delta_{send}^{k,\gamma}}{\Delta_{inter}^{k,\gamma}} \cdot C \cdot \delta + b$$

Computing b

We use the largest class k burst that can be sent with the BLS.

In the presence of MC(k) traffic, the largest period of time during which class k traffic can be sent continuously occurs if the credit started at 0. Then, class k traffic is sent continuously until L_M^k is reached and the priority is changed to its low value p_L. If a new class k frame started its transmission just before the credit reached L_M^k due to non-preemption, it will finish its transmission before the waiting MC(k) traffic can be sent. Thus, with a link capacity C the largest class k burst is $b_k^{max} = \frac{C}{I_{send}^k} \cdot L_M^k + MFS_k$. This gives:

$$\Delta R_k^*(\frac{b_k^{max}}{C}) \leqslant b_k^{max} = \frac{\Delta_{send}^{k,\gamma}}{\Delta_{inter}^{k,\gamma}} \cdot b_k^{max} + b$$

$$\Rightarrow b = b_k^{max} \cdot \frac{\Delta_{idle}^{k,\gamma}}{\Delta_{inter}^{k,\gamma}}$$

So this gives: $\forall \delta \geqslant \frac{b_k^{max}}{C}$, $\Delta R_k^*(\delta) \leqslant \frac{\Delta_{send}^{k,\gamma}}{\Delta_{inter}^{k,\gamma}} \cdot C \cdot \delta + b_k^{max} \cdot \frac{\Delta_{idle}^{k,\gamma}}{\Delta_{inter}^{k,\gamma}}$.

Additionally, we have: $\forall \delta \leqslant \frac{b_k^{max}}{C}$, $\Delta R_k^*(\delta) \leqslant C \cdot \delta \leqslant \frac{\Delta_{send}^{k,\gamma}}{\Delta_{inter}^{k,\gamma}} \cdot C \cdot \delta + b_k^{max} \cdot \frac{\Delta_{idle}^{k,\gamma}}{\Delta_{inter}^{k,\gamma}}$.

So, we have proved that $\forall \delta \in \mathbb{R}^+$,

$$\Delta R_k^*(\delta) \leqslant \frac{\Delta_{send}^{k,\gamma}}{\Delta_{inter}^{k,\gamma}} \cdot C \cdot \delta + b_k^{max} \cdot \frac{\Delta_{idle}^{k,\gamma}}{\Delta_{inter}^{k,\gamma}}.$$

8.6 Window-based vs Continuous Credit-based approaches when $L_R = 0$ for 3-classes case study

Consider the case study defined in Chapter 3. As there is only one shaped class: SCT, we use $k = \emptyset$ to simplify the notation whenever possible.

Concerning the modelisation of the strict minimum service curve, the initial latency is identical: $\Delta^{max}_{idle} = \frac{L_M - L_R}{I_{idle}} + \frac{MFS_{RC}}{C} = \Delta^{\beta}_{idle}$.

In the specific case where $L_R = 0$, both models have also the same strict minimum service curve rate $R^{bls}_{\beta,SCT}(L_R = 0)$:

$$\begin{aligned}
R^{bls}_{\beta,SCT}(L_R = 0) &= \left(1 - \frac{MFS^{max}_{RC,sat}(L_R = 0)}{\Delta^{\beta}_{inter}(L_R = 0) \cdot C}\right) \cdot I_{idle} \\
&= \left(1 - \frac{\frac{MFS_{RC}}{C}}{\frac{L_M}{I_{send}} + \frac{L_M}{I_{idle}} + \frac{MFS_{RC}}{C}}\right) \cdot I_{idle}
\end{aligned}$$

We now use the definition of $I_{idle} = BW \cdot C$ and $I_{send} = (1 - BW) \cdot C$:

$$\begin{aligned}
R^{bls}_{\beta,SCT}(L_R = 0) &= \left(\frac{\frac{L_M}{BW \cdot C} + \frac{L_M}{(1-BW)\cdot C}}{\frac{L_M}{I_{send}} + \frac{L_M}{I_{idle}} + \frac{MFS_{RC}}{C}}\right) \cdot BW \cdot C \\
&= \left(\frac{\frac{L_M \cdot (1-BW)}{(1-BW)\cdot C} + \frac{L_M}{(1-BW)\cdot C}}{\frac{L_M}{I_{send}} + \frac{L_M}{I_{idle}} + \frac{MFS_{RC}}{C}}\right) \cdot C \\
&= \frac{\frac{L_M}{(1-BW)\cdot C}}{\frac{L_M}{I_{send}} + \frac{L_M}{I_{idle}} + \frac{MFS_{RC}}{C}} \cdot C
\end{aligned}$$

Finally, we find the expression of the rate of the strict minimum curve with the window-based model.

$$\begin{aligned}
R^{bls}_{\beta,SCT}(L_R = 0) &= \frac{\frac{L_M}{I_{send}}}{\frac{L_M}{I_{send}} + \frac{L_M}{I_{idle}} + \frac{MFS_{RC}}{C}} \cdot C \\
&= \frac{\Delta^{min}_{send}(L_R = 0)}{\Delta^{min}_{send}(L_R = 0) + \Delta^{max}_{idle}(L_R = 0)} \cdot C
\end{aligned}$$

Concerning the maximum service curve, in the specific case where $L_R = 0$, both models have also the same service curve:

$$\begin{aligned}
\gamma_{SCT}(t, L_R = 0) &= \frac{\Delta^{\gamma,send}_{inter}(L_R = 0)}{\Delta^{\gamma}_{inter}(L_R = 0)} \cdot C \cdot t + b^{max}_{SCT}(L_R = 0) \cdot \frac{\Delta^{\gamma,idle}_{inter}(L_R = 0)}{\Delta^{\gamma}_{inter}(L_R = 0)} \\
&= \frac{\frac{L_M}{I_{send}} + \frac{MFS_{SCT}}{C}}{\frac{L_M}{I_{send}} + \frac{L_M}{I_{idle}} + \frac{MFS_{SCT}}{C}} \cdot C \cdot t
\end{aligned}$$

$$
\begin{aligned}
& +\left(\frac{L_M}{I_{send}} \cdot C + MFS_{SCT}\right) \cdot \frac{\frac{L_M}{I_{idle}}}{\frac{L_M}{I_{send}} + \frac{L_M}{I_{idle}} + \frac{MFS_{SCT}}{C}} \\
= \quad & \frac{\Delta_{send}^{max}(L_R = 0)}{\Delta_{send}^{max}(L_R = 0) + \Delta_{idle}^{min}(L_R = 0)} \cdot C \cdot t \\
& +\Delta_{send,0}^{max}(L_R = 0) \cdot C \cdot \frac{\Delta_{idle}^{min}(L_R = 0)}{\Delta_{send}^{max}(L_R = 0) + \Delta_{idle}^{min}(L_R = 0)}
\end{aligned}
$$

Thus, for $L_R = 0$, the two models are identical.

REFERENCES

[1] Airlines Electronic Engineering Committee. Aircraft Data Network Part 7, Avionics Full Duplex Switched Ethernet (AFDX) Network, ARINC Specification 664. Aeronautical Radio, 2002.

[2] R. Bosch GmbH. CAN specification Version 2,0. Technical report, 1991.

[3] Condor Engineering. MIL-STD-1553 Tutorial.

[4] C. M. Fuchs and others. The evolution of avionics networks from ARINC 429 to AFDX. *Network Architectures and Services*, 2012.

[5] H. Ayed, A. Mifdaoui, and C. Fraboul. Interconnection optimization for multi-cluster avionics networks. In *Real-Time Systems (ECRTS), 2013 25th Euromicro Conference on*, pages 145–154. IEEE, 2013.

[6] C. R. Spitzer. Avionics: Elements, Software and Functions. CRC Press, 2016.

[7] C. B. Watkins and R. Walter. Transitioning from federated avionics architectures to integrated modular avionics. In *Digital Avionics Systems Conference, 2007. DASC'07. IEEE/AIAA 26th*, pages 2–A. IEEE, 2007.

[8] Y. Isik. ARINC 629 data bus standard on aircrafts. *Recent Researches in Circuits, Systems, Electronics, Control & Signal Processing*, pages 191–195, 2010.

[9] P. Vdovin and V. Kostenko. Organizing message transmission in AFDX networks. *Programming and Computer Software*, 43(1):1–12, 2017.

[10] M. Li, G. Zhu, Y. Savaria, and M. Lauer. Reliability enhancement of redundancy management in AFDX networks. *IEEE Transactions on Industrial Informatics*, 13(5):2118–2129, 2017.

[11] P. J. Prisaznuk. ARINC 653 role in integrated modular avionics (IMA). In *Digital Avionics Systems Conference, 2008. DASC 2008. IEEE/AIAA 27th*, pages 1–E. IEEE, 2008.

[12] S. International. ARP4761. In *Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment*, 1996.

[13] A. Burns and R. Davis. Mixed criticality systems-a review. *Department of Computer Science, University of York, Tech. Rep*, pages 1–69, 2013.

[14] O. Cros, L. George, and X. Li. A protocol for mixed-criticality management in switched ethernet networks. In *Proc. 3rd Workshop on Mixed Criticality Systems (WMC), RTSS, L. Cucu-Grosjean and R. Davis (Eds.)*, pages 12–17, 2015.

[15] F.-J. Gotz. Traffic Shaper for Control Data Traffic (CDT). *IEEE 802 AVB Meeting*, 2012.

[16] A. Demers, S. Keshav, and S. Shenker. Analysis and simulation of a fair queueing algorithm. In *ACM SIGCOMM Computer Communication Review*, volume 19, pages 1–12. ACM, 1989.

[17] W. Steiner, G. Bauer, B. Hall, M. Paulitsch, and S. Varadarajan. TTEthernet Dataflow Concept. In *Eighth IEEE International Symposium on Network Computing and Applications*, 2009. doi: 10.1109/NCA.2009.28.

[18] H.-T. Lim, D. Herrscher, M. J. Waltl, and F. Chaari. Performance analysis of the ieee 802.1 ethernet audio/video bridging standard. In *Proceedings of the 5th International ICST Conference on Simulation Tools and Techniques*, pages 27–36. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2012.

[19] M. Andrews. Probabilistic end-to-end delay bounds for earliest deadline first scheduling. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 603–612. IEEE, 2000.

[20] H. M. Goldberg. Analysis of the earliest due date scheduling rule in queueing systems. *Mathematics of Operations Research*, 2(2):145–154, 1977.

[21] IEEE TSN Task Group. TSN Specifications. URL `http://www.ieee802.org/1/pages/tsn.html`.

[22] H. Kopetz, A. Ademaj, P. Grillinger, and K. Steinhammer. The time-triggered ethernet (tte) design. In *Object-Oriented Real-Time Distributed Computing, 2005. ISORC 2005. Eighth IEEE International Symposium on*, pages 22–33. IEEE, 2005.

[23] N. Audsley, A. Burns, M. Richardson, K. Tindell, and A. J. Wellings. Applying new scheduling theory to static priority pre-emptive scheduling. *Software Engineering Journal*, 8(5):284–292, 1993.

[24] H. Zhang and D. Ferrari. Rate-controlled static-priority queueing. In *INFOCOM'93. Proceedings. Twelfth Annual Joint Conference of the IEEE Computer and Communications Societies. Networking: Foundation for the Future, IEEE*, pages 227–236. IEEE, 1993.

[25] K. W. Tindell, A. Burns, and A. J. Wellings. An extendible approach for analyzing fixed priority hard real-time tasks. *Real-Time Systems*, 6(2):133–151, 1994.

[26] B. Andersson, S. Baruah, and J. Jonsson. Static-priority scheduling on multiprocessors. In *Real-Time Systems Symposium, 2001.(RTSS 2001). Proceedings. 22nd IEEE*, pages 193–202. IEEE, 2001.

[27] J. Grieu. *Analyse et évaluation de techniques de commutation Ethernet pour l'interconnexion des systèmes avioniques*. PhD thesis, INPT, 2004.

[28] H. Charara, J.-L. Scharbarg, J. Ermont, and C. Fraboul. Methods for bounding end-to-end delays on an AFDX network. In *Real-Time Systems, 2006. 18th Euromicro Conference on*, pages 10–pp. IEEE, 2006.

[29] M. Anand, S. Vestal, S. Dajani-Brown, and I. Lee. Formal modeling and analysis of the AFDX frame management design. In *Object and Component-Oriented Real-Time Distributed Computing, 2006. ISORC 2006. Ninth IEEE International Symposium on*, pages 7–pp. IEEE, 2006.

[30] H. Bauer, J.-L. Scharbarg, and C. Fraboul. Improving the worst-case delay analysis of an AFDX network using an optimized trajectory approach. *IEEE Transactions on Industrial informatics*, 6(4):521–533, 2010.

[31] 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*, 48(1):101–133, 2012.

[32] X. Li, O. Cros, and L. George. The trajectory approach for afdx fifo networks revisited and corrected. In *Embedded and Real-Time Computing Systems and Applications (RTCSA), 2014 IEEE 20th International Conference on*, pages 1–10. IEEE, 2014.

[33] J.-L. Scharbarg, F. Ridouard, and C. Fraboul. A probabilistic analysis of end-to-end delays on an AFDX avionic network. *IEEE transactions on industrial informatics*, 5(1): 38–49, 2009.

[34] F. Ridouard, J.-L. Scharbarg, and C. Fraboul. Probabilistic upper bounds for heterogeneous flows using a static priority queueing on an AFDX network. In *Emerging Technologies and Factory Automation, 2008. ETFA 2008. IEEE International Conference on*, pages 1220–1227. IEEE, 2008.

[35] A. Wierman and M. Harchol-Balter. Classifying scheduling policies with respect to unfairness in an m/gi/1. In *ACM SIGMETRICS Performance Evaluation Review*, volume 31, pages 238–249. ACM, 2003.

[36] J. C. Bennett and H. Zhang. WF2Q: worst-case fair weighted fair queueing. In *INFOCOM'96. Fifteenth Annual Joint Conference of the IEEE Computer Societies. Networking the Next Generation. Proceedings IEEE*, volume 1, pages 120–128. IEEE, 1996.

[37] J. L. Rexford, A. G. Greenberg, and F. G. Bonomi. Hardware-efficient fair queueing architectures for high-speed networks. In *INFOCOM'96. Fifteenth Annual Joint Conference*

of the IEEE Computer Societies. Networking the Next Generation. Proceedings IEEE, volume 2, pages 638–646. IEEE, 1996.

[38] T. ZHOU and X. Huagang. Design of energy-efficient hierarchical scheduling for integrated modular avionics systems. *Chinese Journal of Aeronautics*, 2012.

[39] Y. Hua and X. Liu. Scheduling design and analysis for end-to-end heterogeneous flows in an avionics network. In *INFOCOM, 2011 Proceedings IEEE*, pages 2417–2425. IEEE, 2011.

[40] S. Madhavapeddi and V. George. Efficient fair queuing using deficit round-robin. *Networking, IEEE/ACM Transactions on*, 1996.

[41] H. Wang, C. Shen, and K. G. Shin. Adaptive-weighted packet scheduling for premium service. In *Communications, 2001. ICC 2001. IEEE International Conference on*, volume 6, pages 1846–1850. IEEE, 2001.

[42] L. Ji, T. Arvanitis, and S. Woolley. Fair weighted round robin scheduling scheme for DiffServ networks. *Electronics Letters*, 39(3):333–335, 2003.

[43] L. Lenzini, E. Mingozzi, and G. Stea. Tradeoffs between low complexity, low latency, and fairness with deficit round-robin schedulers. *IEEE/ACM Transactions on Networking (TON)*, 2004.

[44] D. Thiele, J. Diemer, P. Axer, R. Ernst, and J. Seyler. Improved formal worst-case timing analysis of weighted round robin scheduling for ethernet. In *Hardware/Software Codesign and System Synthesis (CODES+ ISSS), 2013 International Conference on*, pages 1–10. IEEE, 2013.

[45] M. Boyer, G. Stea, and W. M. Sofack. Deficit Round Robin with network calculus. In *Performance Evaluation Methodologies and Tools (VALUETOOLS)*, 2012.

[46] H. Yu and L. Xue. Scheduling heterogeneous flows with delay-aware deduplication for avionics applications. *IEEE Transactions on Parallel and Distributed Systems*, 2012.

[47] M. Abuteir and R. Obermaisser. Simulation environment for time-triggered ethernet. In *Industrial Informatics (INDIN), 2013 11th IEEE International Conference on*, pages 642–648. IEEE, 2013.

[48] T. Steinbach, H. D. Kenfack, F. Korf, and T. C. Schmidt. An extension of the OMNeT++ INET framework for simulating real-time ethernet with high accuracy. In *Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques*, pages 375–382. ICST, 2011.

[49] Z. Luxi, P. Paul, L. Qiao, C. Junyan, and X. Huagang. Timing analysis of rate-constrained traffic in TTEthernet using network calculus. *Real-Time Systems*, 2017.

[50] D. TamasSelicean, P. Pop, and W. Steiner. Timing analysis of rate constrained traffic for the ttethernet communication protocol. In *Real-Time Distributed Computing (ISORC), 2015 IEEE 18th International Symposium on*, pages 119–126. IEEE, 2015.

[51] W. Steiner. An evaluation of SMT-based schedule synthesis for time-triggered multi-hop networks. In *Real-Time Systems Symposium (RTSS), 2010 IEEE 31st*, pages 375–384. IEEE, 2010.

[52] G. Alderisi, A. Caltabiano, G. Vasta, G. Iannizzotto, T. Steinbach, and L. L. Bello. Simulative assessments of ieee 802.1 ethernet avb and time-triggered ethernet for advanced driver assistance systems and in-car infotainment. In *Vehicular Networking Conference (VNC), 2012 IEEE*, pages 187–194. IEEE, 2012.

[53] J. Imtiaz, J. Jasperneite, and L. Han. A performance study of Ethernet Audio Video Bridging (AVB) for Industrial real-time communication. In *Emerging Technologies & Factory Automation, 2009. ETFA 2009. IEEE Conference on*, pages 1–8. IEEE, 2009.

[54] J. Jasperneite, J. Imtiaz, M. Schumacher, and K. Weber. A proposal for a generic real-time ethernet system. *IEEE Transactions on Industrial Informatics*, 5(2):75–85, 2009.

[55] E. Heidinger, F. Geyer, S. Schneele, and M. Paulitsch. A performance study of Audio Video Bridging in aeronautic Ethernet networks. In *Industrial Embedded Systems (SIES), 2012 7th IEEE International Symposium on*, pages 67–75. IEEE, 2012.

[56] S. Schneele and F. Geyer. Comparison of IEEE AVB and AFDX. In *Digital Avionics Systems Conference (DASC), 2012 IEEE/AIAA 31st*, pages 7A1–1. IEEE, 2012.

[57] D. Azua, J. A. Ruiz, and others. Complete modelling of AVB in network calculus framework. In *Proceedings of the 22nd International Conference on Real-Time Networks and Systems*, page 55. ACM, 2014.

[58] X. Li and L. George. Deterministic delay analysis of avb switched ethernet networks using an extended trajectory approach. *Real-Time Systems*, 53(1):121–186, 2017.

[59] F. Reimann, S. Graf, F. Streit, M. Glaß, and J. Teich. Timing analysis of Ethernet AVB-based automotive E/E architectures. In *Emerging Technologies & Factory Automation (ETFA), 2013 IEEE 18th Conference on*, pages 1–8. IEEE, 2013.

[60] D. T. J.Diemer and R. Ernst. Formal worst-case timing analysis of Ethernet topologies with strict-priority and AVB switching. In *SIES*, 2012.

[61] U. D. Bordoloi, A. Aminifar, P. Eles, and Z. Peng. Schedulability analysis of ethernet avb switches. In *Embedded and Real-Time Computing Systems and Applications (RTCSA), 2014 IEEE 20th International Conference on*, pages 1–10. IEEE, 2014.

[62] M. Ashjaei, G. Patti, M. Behnam, T. Nolte, G. Alderisi, and L. L. Bello. Schedulability analysis of ethernet audio video bridging networks with scheduled traffic support. *Real-Time Systems*, 53(4):526–577, 2017.

[63] J. Imtiaz, J. Jasperneite, and K. Weber. Approaches to reduce the latency for high priority traffic in IEEE 802.1 AVB networks. In *Factory Communication Systems (WFCS), 2012 9th IEEE International Workshop on*, pages 161–164. IEEE, 2012.

[64] Z. Zhou, Y. Yan, S. Ruepp, and M. Berger. Analysis and implementation of packet preemption for time sensitive networks. In *High Performance Switching and Routing (HPSR), 2017 IEEE 18th International Conference on*, pages 1–6. IEEE, 2017.

[65] D. Thiele, R. Ernst, and J. Diemer. Formal worst-case timing analysis of Ethernet TSN's time-aware and peristaltic shapers. In *VNC*. IEEE, 2015.

[66] S. Thangamuthu, N. Concer, P. J. 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*, pages 55–60. IEEE, 2015.

[67] S. S. Craciunas, R. S. Oliver, and W. Steiner. Formal scheduling constraints for time-sensitive networks. *arXiv preprint arXiv:1712.02246*, 2017.

[68] D. Maxim and Y.-Q. Song. Delay Analysis of AVB traffic in Time-Sensitive Networks (TSN). In *RTNS 2017-International Conference on Real-Time Networks and Systems*, page 10, 2017.

[69] S. Brunner, J. Roder, M. Kucera, and T. Waas. Automotive E/E-architecture enhancements by usage of ethernet TSN. In *Intelligent Solutions in Embedded Systems (WISES), 2017 13th Workshop on*, pages 9–13. IEEE, 2017.

[70] N. Navet, J. Villanueva, J. Migge, and M. Boyer. Insights on the performance and configuration of avb and tsn in automotive applications. 2017.

[71] M. K. Al-Hares, P. Assimakopoulos, D. Muench, and N. J. Gomes. Traditional queuing regimes and time-aware shaping performance comparison in an Ethernet fronthaul network. In *Transparent Optical Networks (ICTON), 2017 19th International Conference on*, pages 1–4. IEEE, 2017.

[72] D. Hisano, Y. Nakayama, T. Kubo, T. Shimizu, H. Nakamura, J. Terada, and A. Otaka. Gate-Shrunk Time Aware Shaper: Low-Latency Converged Network for 5G Fronthaul and M2M Services. In *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pages 1–6. IEEE, 2017.

[73] M. K. Al-Hares, P. Assimakopoulos, D. Muench, and N. J. Gomes. Modeling Time Aware Shaping in an Ethernet Fronthaul. In *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pages 1–6. IEEE, 2017.

[74] M. J. Teener. Back to the future:using TAS and preemption for deterministic distributed delays. *IEEE 802.1 AVB TG Meeting, San Antonio*, 2012.

[75] D. Thiele and R. Ernst. Formal worst-case timing analysis of Ethernet TSN's burst-limiting shaper. In *DATE*, 2016.

[76] J. Specht and S. Samii. Urgency-based scheduler for time-sensitive switched ethernet networks. In *Real-Time Systems (ECRTS), 2016 28th Euromicro Conference on*, pages 75–85. IEEE, 2016.

[77] J.-Y. L. Boudec. A theory of traffic regulators for deterministic networks with application to interleaved regulators. *arXiv preprint arXiv:1801.08477*, 2018.

[78] J. Specht and S. Samii. Synthesis of queue and priority assignment for asynchronous traffic shaping in switched ethernet. In *Real-Time Systems Symposium (RTSS), 2017 IEEE*, pages 178–187. IEEE, 2017.

[79] V. M. 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*, 2017.

[80] W. Steiner, P. Heise, and S. Schneele. Recent ieee 802 developments and their relevance for the avionics industry. In *2014 IEEE/AIAA 33rd DASC*. doi: 10.1109/DASC.2014.6979419.

[81] R. Coelho, G. Fohler, and J.-L. Scharbarg. Worst-case backlog for AFDX network with n-priorities. In *RTN*, 2014.

[82] Standard. IEEE Std. 802.1Qav, IEEE Standard for Local and metropolitan area networks, Virtual Bridged Local Area Networks, Amendment 12: Forwarding and Queuing Enhancements for Time-Sensitive Streams, 2009.

[83] IEEE. 802.1Q - Virtual LANs. URL `http://www.ieee802.org/1/pages/802.1Q.html`.

[84] F. Baker, J. Polk, and M. Dolly. A differentiated services code point (dscp) for capacity-admitted traffic, May 2010. URL `http://tools.ietf.org/rfc/rfc5865.txt`. RFC5865.

[85] D. Black and P. Jones. Differentiated services (diffserv) and real-time communication, November 2015. URL `http://tools.ietf.org/rfc/rfc7657.txt`. RFC7657.

[86] F.-J. G. S. Kerschbaum and F. Chen. Towards the Calculation of Performance Guarantees for BLS in Time-Sensitive Networks. *IEEE 802.1 TSN Meeting*, 2013.

[87] J. Le Boudec and P. Thiran. *Network calculus: a theory of deterministic queuing systems for the internet*. Springer-Verlag, 2001.

[88] P. Simon, W. Ernesto, T. Lothar, and et al. Influence of different abstractions on the performance analysis of distributed hard real-time systems. *Design Automation for Embedded Systems*, 2009. ISSN 1572-8080. doi: 10.1007/s10617-008-9015-1.

[89] J. Loeser and H. Haertig. Low-latency hard real-time communication over switched ethernet. In *ECRTS*. IEEE, 2004.

[90] M. Fidler, V. Sander, and W. Klimala. Traffic shaping in aggregate-based networks: implementation and analysis. *Computer Communications*, 2005.

[91] A. Bouillard, L. Jouhet, and E. Thierry. Service curves in Network Calculus: dos and don'ts. Research report, INRIA, 2009.

[92] M. Boyer. Half-modeling of shaping in FIFO net with network calculus. In *18th International Conference on Real-Time and Network Systems*, pages 59–68, Toulouse, France, November 2010.

[93] **A. Finzi**, A. Mifdaoui, E. Lochin, and F. Frances. Mixed-Criticality on the AFDX Network: Challenges and Potential Solutions. In *ERTS*, 2017.

[94] **A. Finzi**, A. Mifdaoui, E. Lochin, and F. Frances. Network Calculus-based Timing Analysis of AFDX networks with Strict Priority and TSN/BLS Shapers. In *SIES*, 2018.

[95] **A. Finzi**, A. Mifdaoui, E. Lochin, and F. Frances. Incorporating TSN/BLS in AFDX for Mixed-Criticality Applications: Model and Timing Analysis. In *WFCS*, 2018.

[96] **A. Finzi**, A. Mifdaoui, E. Lochin, and F. Frances. Performance Enhancement of Extended AFDX via Bandwidth Reservation for TSN/BLS Shapers. In *RTN*, 2018.

[97] **A. Finzi**, E. Lochin, A. Mifdaoui, and F. Frances. Improving RFC5865 Core Network Scheduling with a Burst Limiting Shaper. In *IETF*, 2017.

[98] **A. Finzi** and E. Lochin. Improving RFC5865 Core Network Scheduling with a Burst Limiting Shaper. URL `https://datatracker.ietf.org/meeting/99/materials/slides-99-iccrg-iccrg-presentation-5`.

[99] F. Baker, **A. Finzi**, E. Lochin, A. Mifdaoui, and F. Frances. Priority Switching Scheduler. URL `https://datatracker.ietf.org/doc/draft-finzi-priority-switching-scheduler/`.

[100] J. B. Schmitt, F. A. Zdarsky, and I. Martinovic. Improving performance bounds in feed-forward networks by paying multiplexing only once. In *Measuring, Modelling and Evaluation of Computer and Communication Systems (MMB)*, 2008.

[101] B. Lantz, B. Heller, and N. McKeown. A network in a laptop: rapid prototyping for software-defined networks. In *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks*, page 19. ACM, 2010.

[102] N. G. Nayak, F. Dürr, and K. Rothermel. Incremental Flow Scheduling & Routing in Time-sensitive Software-defined Networks. *IEEE Transactions on Industrial Informatics*, 2017.

[103] Z. Li, Q. Li, L. Zhao, and H. Xiong. Openflow channel deployment algorithm for software-defined afdx. In *Digital Avionics Systems Conference (DASC), 2014 IEEE/AIAA 33rd*, pages 4A6–1. IEEE, 2014.

[104] P. Axer, D. Thiele, R. Ernst, and J. Diemer. Exploiting shaper context to improve performance bounds of ethernet avb networks. In *Proceedings of the 51st Annual Design Automation Conference*, pages 1–6. ACM, 2014.