

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par :

Institut National Polytechnique de Toulouse (Toulouse INP)

Discipline ou spécialité :

Informatique et Télécommunication

Présentée et soutenue par :

M. AAKASH SONI

le vendredi 5 juin 2020

Titre :

Real-time performance analysis of a QoS based industrial embedded network

---

Ecole doctorale :

Mathématiques, Informatique, Télécommunications de Toulouse (MITT)

Unité de recherche :

Institut de Recherche en Informatique de Toulouse (IRIT)

Directeur(s) de Thèse :

M. JEAN-LUC SCHARBARG

M. JEROME ERMONT

Rapporteurs :

M. LAURENT GEORGE, UNIVERSITE MARNE LA VALLEE

M. YE-QIONG SONG, UNIVERSITÉ LORRAINE

Membre(s) du jury :

Mme CHRISTINE ROCHANGE, UNIVERSITE TOULOUSE 3, Président

M. ASSIA SOUKANE, EC INGENIEURS VILLE DE PARIS, Membre

M. FRÉDÉRIC RIDOUARD, ENSMA POITIERS, Membre

M. JEAN-LUC SCHARBARG, TOULOUSE INP, Membre

M. JEROME ERMONT, TOULOUSE INP, Membre

M. LEANDRO SOARES INDRUSIAK, UNIVERSITY OF YORK, Membre

À cœur vaillant rien d'impossible.

— Jacques Cœur (l'auteur original)

— Maria Passariello (pour moi)

Acknowledgements

I love not man the less but Nature more... So, first I thank you the Beloved Nature for providing me years of solace, amusement, wonder and awe. Your beauty and magnificence lift my spirit and gave me a tranquil heart.

This Ph.D. thesis is the result of work whereby I have been accompanied and supported by many people scientifically and socially. It is my pleasure now to express my thanks to all those people who made this thesis possible and an unforgettable experience for me.

I would like to thank my supervisors who trusted me from the start and who knew how to be present at every stage of this adventure. My humble gratitude to my thesis supervisor Mr. Jean-Luc Scharbarg who has been a constant source of inspiration and the guiding light for all my endeavours. I am deeply indebted and sincerely grateful to him for his never-failing direction and encouragement. His passion towards the research and unostentatious nature has always been an inspiration for me. I am grateful to my co-supervisor Mr. Jérôme Ermont for his constant help, motivation, inspiration and stimulating suggestions. His humorous nature, gentle and pleasing personality made the environment live and most productive. I feel privileged to be associated with such great philosophers of Science.

I am extremely obliged to Mrs. Xiaoting Li and Mr. Christian Fraboul who co-supervised my work during the the first half of my thesis research and did not hesitate to accompanied me in my explorations, down to their most technical extremes. Thank you to each of them for their patience and their encouragement.

My sincere appreciation to Mr. Ye-Qiong Song and to Mr. Laurent George who accepted the heavy task of being the reviewers of this thesis. Thank you for all the interest and the valuable feedback that you have given to my research. My deep regards to Mrs Christine Rochange, who agreed to chair my thesis committee, and to the remaining members of my thesis committee Mr Frédéric Ridouard, Mrs Assia Soukane and Mr Leandro Soares Indrusiak. I thank you for the interest you have shown in my work and also for your significant observations.

I won't forget to thank all my colleagues from IRIT/ENSEEIHT Toulouse and ECE

Paris. My dear fellow researchers and doctoral students with whom I had a lot of fun to work : Oana, Adrien, Mohamed, Cédric, Amal, Kévin, Santiago, Elene, Dorin, Romain, Justin, Yousouf, Firmin, Guillaume, Quentin, Thomas, Rafik, Sebti, Manolo, Serena, and Ben. (Please excuse me if I forgot someone) Thank you for being supportive about my work and for providing a nice environment to work and share ideas. Thank you to our dear Sylvie and Muriel who have been flawlessly supportive and extremely effective in overcoming all administrative difficulties. You are precious to the whole team.

My acknowledgement will never be complete without the special mention of seniors who provided an excellent research culture. I would like to acknowledge Mr. André-Luc, Mr. Urtzi, Mrs Katia, Mr. Emmanuel, Mr. Gentian, Mr. Riadh, Mrs Sandrine, Mrs Béatrice and Mr. Julien for their support and motivation during the days of my stay at IRIT.

I would like to acknowledge my friends, who are more like a family to me, for their moral support and motivation, which drives me to give my best Mihael, Shreyas, Hermine, Krishanu, Camille, Elise, Gagny, Amine, Rebecca, the list is endless ... thanks to one and all. I feel proud to have friends like you.

Finally, I would like to acknowledge the people who mean the world to me, my brother for his encouragement and above all, belief in me. My parents for their emotional support and motivation. I am also grateful to my loved ones: Ritika, Manu and Christy, offering unconditional love and support. The present work is an outcome of all their blessings and moral support. I wish to sincerely acknowledge Maria, for being there by my side with compassionate support, lots of love and awesome rhythm.

Thank you all for being in my life.

Abstract

Avionics Full Duplex (AFDX) switched Ethernet network has become the de facto standard for communications of critical avionic data. The certification process of AFDX network requires that the service provided by the network must guarantee the determinism of communications. It relates to the guarantee of the maximum transmission delay of a data packet from one end of the network to the other end. The network is then considered deterministic if it is possible to guarantee that, for each flow, the fixed delay bound is never exceeded. One technique to obtain such bounds is WCTT analysis, which can be a complex process for large network configurations. A reliable and scalable solution is provided by Network Calculus approach which computes pessimistic delay upper bound. This approach has also been used to dimension the network elements, to ensure that the queues are able to store all backlogged traffic. But the pessimism introduced by Network Calculus approach leads to over-dimensioning of the network resources, which in turn leads to lightly loaded network.

The manufacturers envision to better use the available network resources by allowing traffic from different functions, both critical and non-critical. However, currently deployed AFDX network do not differentiate between classes of traffic, i.e. all the frames are served in FIFO manner. Such architecture is not at all convenient to allow traffic of mixed criticality. In this context, implementing a QoS mechanism that differentiate multiple classes of traffic is required. DRR and WRR scheduling are the choice of industry for this purpose.

In this thesis we study the WCTT analysis in such QoS-aware network in order to provide tight delay bounds and a method to efficiently allocate resources in the network shared by flows of mixed criticality. We show how the network bandwidth can be efficiently divided among different classes of flow having different delay constraints by tuning the QoS mechanism. We also propose an optimised network calculus approach that mitigates the pessimism involved in delay computation.

Résumé

Le réseau AFDX est devenu la norme de facto pour les transmissions de données avioniques critiques. La certification du réseau AFDX impose que le service fourni par le réseau garantisse le déterminisme des communications. En particulier, le délai maximale de transmission d'un message d'un bout à l'autre du réseau doit être garantie. Le réseau est alors considéré comme déterministe s'il est possible de garantir que, pour chaque flux, la borne fixée n'est jamais dépassée. Une technique pour obtenir de telles bornes est l'analyse de délai pire-cas, qui peut être un processus complexe pour les réseau de grand taille. Une solution fiable et évolutive est fournie par l'approche Network Calculus (NC) qui calcule une borne supérieur pessimiste du délai. Cette approche a également servi à dimensionner tous les éléments du réseau, de sorte que les files d'attente soit capable de stocker toutes les trames, quel que soit le scénario et en particulier le plus défavorable. Mais le pessimisme introduit par NC conduit à un surdimensionnement des ressources du réseau et ainsi à un réseau légèrement chargé.

Les avionneurs envisagent d'augmenter l'utilisation des ressources du réseau AFDX en ajoutant du trafic supplémentaire provenant d'autres fonctions critiques et non critiques. Cependant, le réseau AFDX actuellement déployé ne fait pas de distinction entre les classes de trafic, c'est-à-dire que toutes les trames sont servies de la manière FIFO. Une telle architecture ne permet pas de gérer du trafic de criticité mixte. Dans ce contexte, le déploiement d'un mécanisme de la qualité de service (QoS) qui différencie plusieurs classes de trafic est requis. Ainsi, les industriels ont choisi de mettre en œuvre des méthodes d'ordonnancement telles que DRR et WRR.

Dans cette thèse, nous étudions l'analyse pire cas du délai d'un tel réseau utilisant de mécanisme de QoS afin d'obtenir des bornes de délai plus petite et une méthode pour allouer efficacement les ressources du réseau partagées entre des flux de criticité mixte. Nous montrons comment la bande passante du réseau peut être efficacement divisée entre différentes classes de flux ayant différentes contraintes de délai. Nous proposons également une approche NC optimisé qui atténue le pessimisme impliqué dans le calcul des délais.

Table of contents

Abstract **ix**

Introduction **1**

1 Switched Ethernet and Determinism **7**

1.1 Introduction 8

1.2 Evolution of switched Ethernet as a real-time network 9

1.2.1 Ethernet: a mature technology for general-purpose networks . . . 9

1.2.2 Real-time Ethernet 10

1.3 AFDX like real-time switched Ethernet network 11

1.3.1 Network architecture 12

1.3.2 Flow model 14

1.3.3 Example of real-time switched Ethernet network 15

1.4 Determinism and bounded delays in the network 15

1.4.1 End-to-end delay 16

1.4.2 Worst-case end-to-end delay analysis 19

1.5 Deterministic method of worst-case delay analysis: Network Calculus approach 22

1.5.1 NC basic concepts 23

1.5.2 Modelling a FIFO scheduling based real-time switched Ethernet network in NC approach 25

1.6 Impact of deterministic approaches in dimensioning a network 31

1.7 Solution to improve network resource utilisation: a QoS-aware switched Ethernet 32

1.8 Thesis scope and contributions 39

1.9 Conclusion 41

2 A QoS-Aware Switched Ethernet Network 43
2.1 Introduction . . . 43
2.2 Real-time switched Ethernet network architecture with DRR scheduling . 44
2.3 Overview of DRR scheduling . . . 45
2.3.1 DRR working principle . . . 45
2.3.2 Bandwidth sharing in DRR scheduler . . . 47
2.4 Worst-Case Traversal Time analysis . . . 50
2.4.1 DRR scheduler latency . . . 50
2.4.2 Modelling real-time switched Ethernet network with DRR scheduling in Network Calculus . . . 56
2.5 Conclusion . . . 58

3 Efficient Resource Sharing in a QoS-Aware Switched Ethernet Network 59
3.1 Introduction . . . 59
3.2 Assumptions . . . 60
3.3 Delay computation by classical NC approach . . . 61
3.4 Quantum assignment . . . 67
3.4.1 Properties of classical NC approach . . . 68
3.4.2 Optimal Quantum assignment algorithm . . . 74
3.5 Conclusion . . . 78

4 An Optimized NC Approach to WCTT Analysis in a Switched Ethernet Network with DRR Scheduling 79
4.1 Introduction . . . 80
4.2 Classical NC approach . . . 80
4.2.1 Modelling switched Ethernet network with DRR schedulers in classical NC approach . . . 80
4.2.2 Pessimism in classical NC approach . . . 82
4.3 Optimizing NC to reduce pessimism . . . 86
4.3.1 Maximized service of interfering classes . . . 87
4.3.2 Effective maximum load of interfering classes . . . 90
4.3.3 Limitation of the service to the load . . . 91
4.4 Flow scheduling at source end-node . . . 92
4.4.1 Introducing offset . . . 93
4.4.2 Integrating offset in NC arrival curve . . . 96
4.5 Evaluation of optimised NC approach . . . 101

4.6 Conclusion . . . 102

5 Case Study on An Industrial Real-Time Switched Ethernet Network : AFDX Network 103

5.1 Introduction . . . 103
5.2 Context of AFDX network . . . 104
5.2.1 AFDX network and flow model . . . 104
5.2.2 Reference AFDX network . . . 105
5.2.3 Flow differentiation . . . 107
5.3 WCTT analysis based on classical NC approach . . . 108
5.3.1 First-In-First-Out (FIFO) scheduling . . . 108
5.3.2 Static Priority Queuing (SPQ) scheduling . . . 111
5.3.3 Deficit Round Robin (DRR) scheduling . . . 111
5.4 WCTT analysis based on optimised NC approach . . . 114
5.5 Conclusion . . . 116

6 WCTT Analysis in a Switched Ethernet Network with WRR Scheduler 119

6.1 Introduction . . . 119
6.2 Overview of WRR schedulers . . . 120
6.2.1 WRR Algorithm . . . 120
6.2.2 Bandwidth sharing in WRR scheduler . . . 123
6.3 Worst-Case Traversal Time analysis . . . 127
6.3.1 WRR scheduler latency . . . 127
6.3.2 Network Calculus approach . . . 128
6.4 Case study on industrial AFDX network . . . 135
6.4.1 WCTT analysis on an industrial size AFDX network configuration 135
6.4.2 WCTT analysis on a modified configuration of an industrial size AFDX network . . . 139
6.4.2.1 WRR scheduling with homogeneous flows . . . 139
6.4.2.2 WRR scheduling with heterogeneous flows . . . 144
6.5 Conclusion . . . 147

Conclusion 149

Glossary 153

Acronyms 155

Bibliography **157**

List of figures **163**

List of tables **165**

Introduction

General Introduction

Modern real-time embedded networks, such as aeronautic on-board ones, require hard real-time guarantees to ensure safe and timely operations of critical functions. Such guarantees have first been provided by specific fieldbus technologies, like Controller Area Network (CAN) for automotive or ARINC 429 for avionics. However these fieldbuses provide very limited bandwidth and cannot cope with the huge increase of communication needs in real-time embedded systems.

Therefore Ethernet has been considered. However, the traditional Ethernet is based on Carrier Sense Multiple Access with Collision Detection (CSMA/CD) which is not deterministic. Indeed collisions may occur and the collision resolution mechanism cannot guarantee that a frame will be transmitted within a bounded delay.

Real-time Ethernet solutions have been proposed. They eliminate collisions or provide deterministic collision resolution. They can also provide solutions to bound the traffic. A popular real-time Ethernet solution is Avionic Full-Duplex switched Ethernet (AFDX). It fully eliminates collisions thanks to switches and full duplex links. Moreover the bandwidth allocated to each flow is upper bounded, thanks to shaper implemented in each end system. The determinism of communication required by the mandatory certification process can be guaranteed, thanks to Worst Case Traversal Time (WCTT) analysis using Network Calculus. Therefore AFDX became the de facto standard for critical avionic communications.

The WCTT analysis has to take into account very rare scenarios. It makes some pessimistic assumptions, leading to over-dimensioning of the network architecture, therefore to lightly loaded network. For instance, link of a typical A380 AFDX network are loaded at less than 10% on average.

The manufacturers envision to use this available bandwidth by allowing traffic from different functions. Presently, the existing communication network architecture in

aeronautic system employs parallel networks: an AFDX network to provide service only to the high safety level functions in the avionics domain and one or more dedicated networks (based on CAN, ARINC429, Ethernet, ...) for non-critical comfort functions like diagnostic messages, collective maintenance and In-Flight entertainment and for critical functions like inter-cabin communication, smoke detection, parking video, emergency announcements etc. The main advantages of the current architecture are its very high fidelity and an existing certification. Nonetheless, these advantages come at the cost of inefficient resource utilisation, high complexity and difficult maintainability. Moreover, the separate networks require use of different cables, which run largely in parallel. Since reducing weight is a primary objective in aircraft construction, this is an undesirable situation. In this context, the manufacturers intend to blend the parallel networks by integration of various functions into a single AFDX backbone and thus improve the bandwidth utilisation.

Such harmonisation of AFDX network with the data of different criticality levels leads to the question whether the additional data traffic can be served without degrading the service provided to the data of highest criticality. Since the network must provide hard real-time performance guarantees for the safety critical applications, for instance no packet loss is tolerable from the avionic function due to buffer overflow and these data packets are constrained by maximum allowable delays which, in any condition, must be respected to meet the requirements of the certification process.

In absence of a flow differentiation mechanism the addition of less critical or non-critical data, potentially bandwidth consuming, would have consequences in terms of increased delay on existing critical data. This is why the manufacturers envision the network architecture with Quality-of-Service (QoS) mechanism such as Deficit Round Robin (DRR) and Weighted Round Robin (WRR). Such mechanism allows to reserve bandwidth on per data class basis so that a certain level of performance guarantee can be assured for these classes. Moreover, reserving a fair share of bandwidth for different classes can improve the overall performance of the network.

So, in this thesis our goal is to analyse and maximise the performance of such QoS-aware network with mixed-criticality data flow. In particular, we investigate the WCTT analysis in this network in order to provide tight delay bounds and a method to efficiently allocate resources to different classes of flow. In the first step we consider a DRR scheduling based network with (n-1) critical classes and 1 non-critical class. The delay bound guarantee in this network is provided based on Network Calculus approach. Since the DRR scheduling manages resource sharing based on predefined credit (or quantum) per traffic class, we illustrate the impact of these credits on the delays bounds. Then

we propose an algorithm that provides an optimal credit assignments which guarantees that the delay constraints in critical classes are always respected and the bandwidth available to non-critical class is maximum. This problem is of great interest since it can significantly improve the resource utilisation in an industrial setting.

Our algorithm relies on Network Calculus approach for delay bound computation. An important issue with the Network Calculus is the pessimism: it computes an overrated upper bounds rather than exact worst-case delays. This also affects the efficiency of our algorithm. So, in the second step, we illustrate the pessimism introduced in delay bounds by Network Calculus approach. Then we propose some optimisation in Network Calculus model to mitigate the pessimism. In order to quantify the improvements, we evaluate our credit assignment algorithm and optimised Network Calculus approach on an industrial size AFDX network configuration.

In the final step, we consider a WRR scheduling based network. We have illustrated how our optimised Network Calculus approach can be extended for this network. In order to compare the performance of WRR and DRR scheduling in industrial setting we have also performed another case study on an industrial size AFDX network configuration.

Thesis Overview

In the first chapter we present the context of real-time switched Ethernet network and the approach to worst-case end-to-end delay analysis. We also recall the problems addressed by this thesis.

Chapter 2 includes presentation of a QoS-aware switched Ethernet network and a state-of-the-art approach (Network Calculus) for worst-case delay analysis of this network. In this chapter, we consider DRR scheduling at each switch output port.

Chapter 3 focuses on optimising the bandwidth utilisation in a switched Ethernet network with DRR scheduling in presence of flows of mixed criticality. For that purpose, we propose an algorithm of credit assignment in a DRR scheduler. The main goal is to allocate sufficient bandwidth to critical flows and maximum residual bandwidth to non-critical flows.

In chapter 4 we first present the source of pessimism in worst-case delay analysis in Network Calculus approach when a switched Ethernet network with DRR scheduling is considered. Then we propose some optimisations to mitigate this pessimism in order to compute tight delay bounds. We also propose a method to integrate offset in delay computations.

Chapter 5 introduces the Avionics Full-DupleX (AFDX) switched Ethernet network

as an industrial real-time switched Ethernet example. We conduct the evaluations of our credit assignment algorithm and optimised Network Calculus approach on an industrial AFDX configuration. The results are compared with existing approach. We also perform multiple experiments by introducing variable amount of non-critical traffic in the given network configuration.

In chapter 6 we extend our optimised Network Calculus model for worst-case delay analysis in a WRR scheduling based switched Ethernet network. We analyse the effect of frame length variation within traffic classes on worst-case delay computation in the given network. We also compare the performance of WRR scheduling and DRR scheduling under similar network configuration.

List of publications:

[SLSF18b] Optimizing Network Calculus for Switched Ethernet Network with Deficit Round Robin. The classical Network Calculus approach used to compute the delay bounds in DRR based switched Ethernet network is very pessimistic as it does not take into account the traffic of individual classes into account. This pessimism has a significant impact on dimensioning of the network. In this paper, we proposed an optimised Network Calculus approach to mitigate this pessimism.

Published at 39^{th} *IEEE Real-Time Systems Symposium (RTSS-2018)*

[SLSF18a] Integrating Offset in Worst Case Delay Analysis of Switched Ethernet Network With Deficit Round Robin. The flows which are locally synchronised at their source end-node are temporally dependent. The emission times of these flow frames are separated by a fixed offset value which is not considered by the classical Network Calculus approach. The offset integration in Network Calculus approach was already introduced in the context of FIFO scheduling in a switched Ethernet network. In this paper, we have extended this approach in the context of DRR scheduling. A benefit of knowing the offsets is to improve the worst-case delay analysis since it can eliminate some impossible scenarios.

Published at 23^{rd} *IEEE International Conference on Emerging Technologies and Factory Automation (ETFA-2018)*

[SSE19] Quantum Assignment for QoS-Aware AFDX Network with Deficit Round Robin With DRR, the flow set is divided into classes and each class is allocated a quantum. In each DRR service round, transmissions are managed, based on these quanta. Thus, delays are significantly impacted by quanta. In this paper, we have proposed an efficient quantum assignment for a set of critical flow classes and at most one less/non critical flows.

Published at 27^{th} *International Conference on Real-Time Networks and Systems (RTNS-2019)*

[SLSF18c] WCTT analysis of avionics Switched Ethernet Network with WRR Scheduling. Just like DRR scheduling, WRR is also a promising QoS mechanism in industrial applications where guaranteed or differentiated quality of service is required. Implementation of WRR in AFDX network require guarantee on worst-case delay upper bound. In this paper, we applied our optimised Network Calculus approach to compute tight delay bound on WRR scheduling based AFDX network.

Published at 26^{th} *International Conference on Real-Time Networks and Systems (RTNS-2018)*

[SLSF17] Pessimism analysis of network calculus approach on AFDX networks. Since the pessimism in WCTT analysis has a significant impact on the dimensioning of the network, it is important to quantify the pessimism of the computed upper bounds. In this paper, we propose a generic approach to upper bound this pessimism, based on Network Calculus. Being an initial step, this approach is applied to an AFDX network with FP/FIFO scheduling.

Published at 12^{th} *IEEE International Symposium on Industrial Embedded Systems (SIES-2017)*

Chapter 1

Switched Ethernet and Determinism

Contents

1.1 Introduction 8

1.2 Evolution of switched Ethernet as a real-time network 9

1.2.1 Ethernet: a mature technology for general-purpose networks . 9

1.2.2 Real-time Ethernet 10

1.3 AFDX like real-time switched Ethernet network 11

1.3.1 Network architecture 12

1.3.2 Flow model 14

1.3.3 Example of real-time switched Ethernet network 15

1.4 Determinism and bounded delays in the network 15

1.4.1 End-to-end delay 16

1.4.2 Worst-case end-to-end delay analysis 19

1.5 Deterministic method of worst-case delay analysis: Network Calculus approach 22

1.5.1 NC basic concepts 23

1.5.2 Modelling a FIFO scheduling based real-time switched Ethernet network in NC approach 25

1.6 Impact of deterministic approaches in dimensioning a network 31

1.7 Solution to improve network resource utilisation: a QoS-aware switched Ethernet 32

1.8 Thesis scope and contributions 39

1.9 Conclusion 41

1.1 Introduction

In distributed industrial control systems such as avionic and automotive systems specific fieldbuses, like ARINC429 and Controller Area Network (CAN) [Spe91], have been developed in order to guarantee bounded end-to-end communication latencies for data transmitted over the network. The end-to-end delay guarantee is ensured either by the fieldbus architecture or by a worst-case delay analysis. For instance, ARINC429 is a mono-emitter bus that allows guaranteed delivery of avionic data with hard real-time constraints. However, its mono-emitter characteristics require a large number of cables to establish communication between multiple avionic functions. Moreover, such fieldbus technologies offer a limited bandwidth (100 kbit/sec). Today, such fieldbuses are no more sufficient because of the huge increase in communication needs.

Ethernet-based solutions are promising candidates in such contexts since they are based on mature technology and provide high bandwidth (from 100 Mbps to 1 Gbps) as well as decreasing cost thanks to the available off-the-shelf components [SKS11, KZST11, HMVdK13]. They also allow easy integration in a more general network infrastructure. However, the traditional Ethernet is not suitable for real-time applications due to its non-deterministic characteristic. Many solutions have been proposed in order to make Ethernet real-time such as EtherNet/IP, PROFINET, Powerlink, AVB and AFDX etc. Avionics Full-Duplex (AFDX) switched Ethernet has become the de facto standard for avionic communications. In this thesis, such a solution is considered. It is based on the full-duplex switched Ethernet with time-constrained traffic similar to the AFDX network. The deterministic guarantee is provided on this network by the data flow constraints and by upper bounding the end-to-end delays using a deterministic computation technique.

In this chapter, we first summarize the evolution of Ethernet as a real-time solution and then eventually present the architecture of the real-time switched Ethernet network which is taken as a reference network in this thesis. We also give a brief introduction of the deterministic approach used to upper bound the end-to-end delays in this network. At last, we give the direction towards the problems addressed and the contributions conveyed by the work of this thesis.

The organisation of this chapter is as follows: Section 1.2 describes the evolution of switched Ethernet as a real-time solution. Section 1.3 gives the network architecture and the flow model of a real-time switched Ethernet network. Section 1.4 explains the problem of determinism in switched Ethernet networks. The existing approaches for worst-case delay analysis are given in Section 1.5. The impact of the deterministic approach on network design is shown in Section 1.6. Section 1.7 gives an overview of the solutions

envisioned to address existing problems introduced by the deterministic approach on network design. Section 1.8 summarises the scope and problems addressed in this thesis. Section 1.9 concludes this chapter.

1.2 Evolution of switched Ethernet as a real-time network

1.2.1 Ethernet: a mature technology for general-purpose networks

Ethernet was originally developed as a technology for computer networking. A standard Ethernet is based on a shared communication channel and an arbitration mechanism called Carrier Sense Multiple Access with Collision Detection (CSMA/CD). A frame transmitted by a source node on the channel can be received by all the nodes. Since any node can initiate a transmission at any time, collisions may occur. By the CSMA/CD mechanism, a node holds the frame until the shared channel becomes idle. Then it starts transmitting the frame. If two or more nodes start transmission at the same time, a collision occurs, and transmissions are aborted after 64 bytes. Nodes involved in the collision starts retransmission after a random amount of time. This procedure is repeated until a successful transmission (without collision) or a maximum number of retransmissions. Collisions and retransmissions decrease efficiency in terms of bandwidth usage. Moreover, since the time needed for a successful transmission cannot be predicted, the communication is not deterministic.

The amount of collisions highly depends on the traffic and the number of stations in the collision domain, i.e. the shared communication channel. Switched Ethernet limits this collision domain. Indeed, each switch is able to store the frames it receives. Therefore, a collision can occur only between two equipment sharing a link (two switches or a switch and a station). Thus, a collision domain is limited to one link. Implementing full duplex links completely eliminates the collisions.

The Ethernet technology offers scalable and high-speed solutions at a low cost since many Commercial Off-The-Shelf (COTS) products are available. For instance, Ethernet TCP/UDP/IP communication (IEEE 802.3) based on 10GBASE-T cables supports bandwidth up to 10 Gbits/sec. Therefore, Ethernet is more and more considered to cope with the increasing communication needs of real-time embedded applications. However, Ethernet is not real-time. As previously mentioned, the time needed for a successful transmission on shared Ethernet cannot be bounded. Full-Duplex switched Ethernet does not solve this problem. Indeed, it eliminates collisions, but it shifts the problem at the switch level, where buffers might overflow, depending on the traffic. For instance, if a burst of messages destined to the same output arrives at the switch they are stored in

the same buffer where they wait for their turn to be transmitted and since this burst can be too large the buffer can overflow, thus, leading to a loss of messages, which is an undesirable situation in a real-time application.

Therefore, many solutions have been proposed to make Ethernet real-time. In the next paragraph, we briefly summarize the different types of solutions.

1.2.2 Real-time Ethernet

Constraints on communication delays can be very different, depending on the concerned real-time application. First, they can be more or less critical: for some applications such as flight control, missing a delay can have catastrophic consequences, while for some other applications such as video transmission, missing a small and limited percentage of delays is tolerable. In the former case, communications are hard real-time while, in the latter one, they are soft real-time. Second, constraints on delays can be very small (around 1 msec) or very large (seconds). Such very different constraints have led to different real-time Ethernet solutions. A taxonomy of solutions has been proposed in [Dec05].

The first class of solutions implements real-time protocols above the TCP/IP layer without any special modifications. It includes Modbus/TCP [Mod04], EtherNet/IP [Sch01] etc. They utilise existing standards for prioritization and Virtual Local Area Network (VLAN) establishment (IEEE 802.1p and IEEE 802.1Q). Obviously, these solutions are limited to soft real-time communications with long delays (at least 100 msec). Indeed, the TCP/IP protocol implies variable and potentially long delays.

The second class of solutions bypass the TCP/IP protocols and accesses directly to the Ethernet functionality without altering its hardware. These solutions provide real-time guarantees through mechanisms such as master/slave and/or time slicing. It includes Ethernet Powerlink (EPL) [Gro], Time-Critical Control Network (TCnet) [Eth], etc. Such solutions can provide upper bounds on delays. However, these upper bounds cannot be small (in the range of 1 msec), due to COTS features. Therefore, this class of solution can cope with hard real-time communications with constraints in the range of 10 msec.

The third class of solutions uses specific components based on a modified Ethernet mechanism and infrastructure. It includes Profinet IO IRT [Fel04] and EtherCat [Eth04] etc. They have been designed in order to deal with hard real-time constraints in the range of 1 msec. All these solutions have been mainly used in the context of factory automation.

Since the requirements in industrial real-time embedded systems (particularly in automotive and avionics domain) are different from those in factory automation, more

specific solutions like Time-Triggered Ethernet (TTEthernet) [KAGS05, Lov15], Ethernet Audio/Video Bridge (AVB) [GKT14, Bel11, Bel14], Time-Sensitive Network (TSN) [MVNB18, BRKW17] and AFDX are considered.

Avionics Full-Duplex switched Ethernet (AFDX, ARINC 664 [Air05]) is a dedicated solution in the context of aeronautics. It is a switched Ethernet network which has been tailored to take into account avionics constraints:

- specific switches are used, where the switching latency is upper bounded,
- flow routing is fully static,
- the bandwidth allocated to each flow is upper bounded, thanks to a minimum duration between two consecutive frames and a maximum frame size.

Such a network architecture (AFDX like) is considered in this thesis. We present it in the following section, starting with the main features of a real-time switched Ethernet.

1.3 AFDX like real-time switched Ethernet network

A switched Ethernet network (based on IEEE 802.1D standard) implementing full-duplex links provides a collision-free communication. A switch interconnects the input and output nodes to a separate port. Each port is equipped with buffers to store incoming frames and these frames are scheduled based on a service discipline at each output port. Such a technique allows simultaneous transmissions between different nodes. Therefore, it removes the impact of the non-deterministic feature of the original Ethernet inherent in the CSMA/CD arbitration. However, in a full-duplex switched Ethernet, the problem of indeterminism is shifted to the output port of the switch. The frames arriving at a switch input port are forwarded to corresponding output ports based on a forwarding table. The switching process of frames from the input to the output port is very fast and is upper bounded by a known value. In the simplest solution, the frames forwarded to the same output port are buffered in a First-In First-Out (FIFO) queue. Depending on the service discipline there might be more than one queue per output port. Since the frames are served in a first come first serve manner in the output queue, they should wait for a certain amount of time until the frames ahead in the queue are transmitted and the queue can transmit. The waiting time experienced by a frame is variable and depends directly on the instantaneous traffic at the arrival time of the frame in the output port. This uncertainty of waiting delay makes the exact state of the whole network unknown at all times and, thus, the network is non-deterministic at port level. In such full-duplex

switched Ethernet network, if the traffic flow is not controlled a critical frame may be blocked for a long time in a queue and eventually it may miss its deadline. Moreover, in the worst scenario the FIFO queues may overflow which may lead to dropping of frames. It means that this traditional network does not provide the deterministic service by design.

Howbeit, the deterministic guarantee for a real-time application can be provided in this network by employing certain features such as static flow routing and traffic shaping at each flow source that constraints the data traffic entering the network (these are the key features of AFDX network). The architecture and flow model of such a network is presented in following paragraphs.

1.3.1 Network architecture

The network architecture is composed of switches, which are the key elements of architecture, end-nodes, which are the source and destination of data flow, and full-duplex links.

Each switch is a store and forward type (IEEE 802.1d). As previously mentioned, each switch output port has a set of queues managed by a scheduling strategy. The frames arriving at input ports are forwarded to the corresponding output ports based on a static routing table. This forwarding between input and output ports introduce a delay called switching latency. The switching latency is upper bounded by a known value, denoted by sl. The simplest architecture considers one single FIFO queue per output port. This type of scheduling is sufficient for handling a single class of data traffic. To allow flows of mixed criticality, more advanced scheduling has to be considered. Each switch port can be connected to at most one switch or one end-node.

The end-nodes are the source and destination of data traffic in the network. Each end-node generates a sequence of frames, known as a flow, at its output port. Each end-node can manage one or more flows. The buffers in end-node output port are managed by a FIFO scheduling policy. An end-node can be connected to only one switch port. There is no synchronisation between the end-nodes due to the absence of a global clock.

The full-duplex links (IEEE 803.1e) interconnects the switch and end-node ports. The full-duplex characteristic guarantees no collision on links. The maximum bandwidth of links is denoted by$R$.

Figure 1.1 shows an example of such a network architecture. It consists of 4 end-nodes (e_1 – e_4) interconnected by two switches (S_1 – S_2) via full-duplex links forwarding 3 flows (v_1 – v_3). For the simplicity of representation, each full-duplex link is represented by a

unique straight line with an arrowhead pointing towards the flow direction.

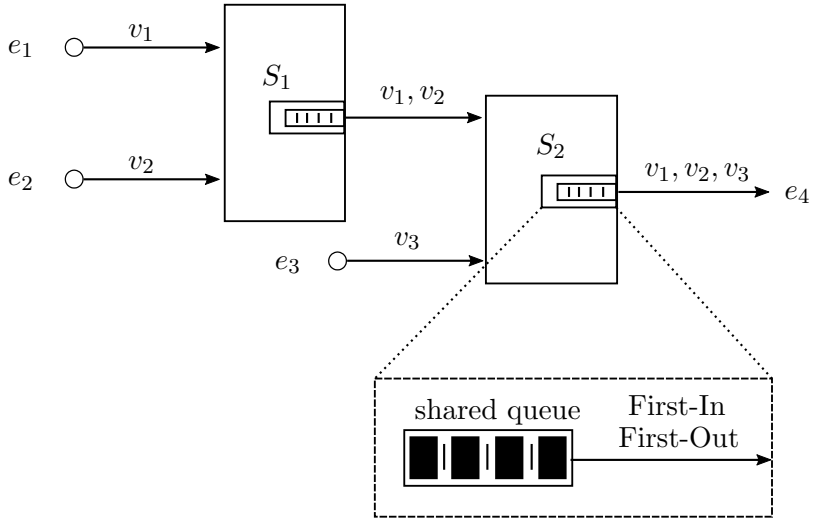


Figure 1.1 An example of a switched Ethernet network

The determinism in this network is partially guaranteed by the following properties:

Static flow configuration The network configuration is static and fully known before deployment. A static configuration eliminates all the network initialization problems as well as the indeterminism related to switching process. This implies that the routing of flow frames are predefined statically and the forwarding tables corresponding to destination MAC address and output ports are statically configured. The static routing avoids any dynamic mechanism such as spanning tree which makes it difficult to evaluate the end-to-end transmission delay in the network.

Traffic shaping and traffic policing The traffic shaping is applied at each end-node to put a restriction on the minimum duration between the emission of two consecutive frames of each flow as well as to limit the size of frames that can be emitted by each flow (traffic contract). In addition, the switches employ traffic policing that detects and eliminates the frames from a flow which does not respect the contract. These two flow control techniques avoid the scenario where a flow could saturate the network resources with continuous frames transmission. This is a key characteristic of a real-time switched Ethernet network since in absence of controlled traffic it would be impossible to upper bound the end-to-end delays.

Store and forward A store-and-forward switch receives a frame and entirely buffers that frame before forwarding it to the related output port. Such a mechanism is essential to eliminate the propagation of erroneous frames by discarding the frames that were not properly received.

1.3.2 Flow model

In this section, we formally define flows. A flow is a sequence of frames emitted by an end-node. Since each end-node can emit one or more flows, a mechanism of flow segregation is provided so that a misbehaving flow can be distinguished from other flows in the network. A flow is defined by the following characteristics:

- A unique ID.
- Destination address. A flow can have one destination (unicast) or many destinations (multicast).
- Static path. A path $\mathcal{P}_i$ followed by a flow v_i is defined as a source node, a sequence of switch output ports and a destination node.
- Minimum (l_i^{min}) and maximum (l_i^{max}) frame lengths. The upper and lower limits on frame length are fixed by the Ethernet (IEEE 802.3) standard as 1518 bytes and 64 bytes.
- Minimum inter-frame arrival time. For each flow v_i, the minimum interval T_i between consecutive frames is fixed at the source end-node by the traffic shaping scheme.

Only unidirectional flows are transmitted in this network. A flow can be sporadic flow or periodic flow. If consecutive frames of a flow are generated at a constant interval which is greater than or equal to its minimum inter-frame arrival time then it is periodic flow, otherwise, it is sporadic flow.

Let us illustrate the difference between a sporadic and a periodic flow with the help of an example shown in Figure 1.2. In this example, three frames f_1, f_2 and f_3 of a flow with minimum inter-frame arrival time T are emitted by an end node. As shown in Figure 1.2, in a periodic flow the generation time ($\downarrow$) of the frames is always separated by a constant interval (in this example it is equal to T) whereas in sporadic flow the separation of frame generation time ($\downarrow$) is non-constant and it can be an interval greater than or equal to T. Besides, for both periodic and sporadic flows, the start of transmission ($\uparrow$) of a frame in output port can be different from its generation time ($\downarrow$) and the difference between the generation time ($\downarrow$) and the start of transmission time ($\uparrow$) is limited by a release jitter j.

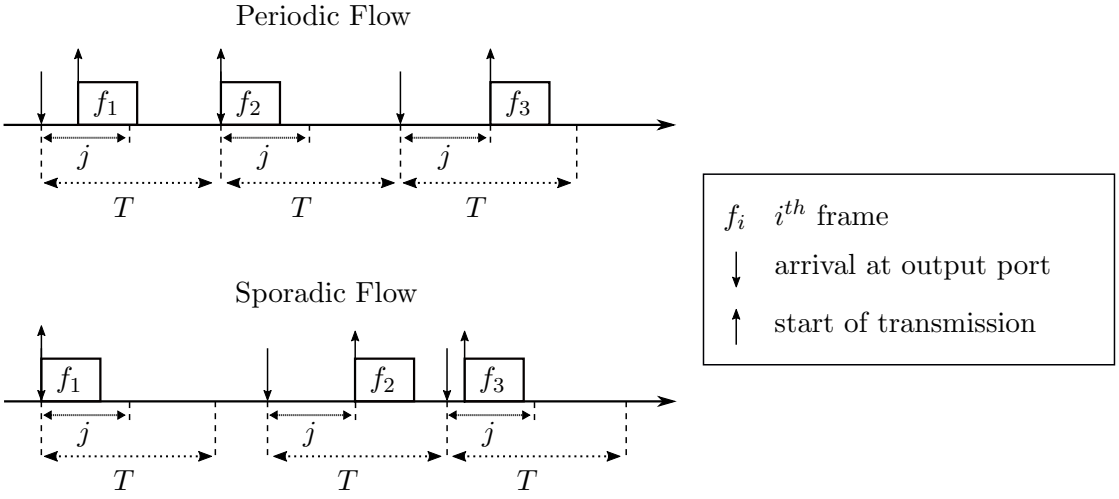


Figure 1.2 Characteristics of sporadic and periodic flows

1.3.3 Example of real-time switched Ethernet network

A typical example of the real-time switched Ethernet presented in the previous paragraphs is Avionics Full DupleX Switched Ethernet (AFDX) [Air05]. AFDX network was developed to address the real-time constraints in modern aircraft such as Airbus A380. An AFDX network consists of end-systems, which are the source and destination of data flows, interconnected by AFDX switches. The statically defined flows transmitted on this network are called Virtual Links. These flows are constrained at their end-system by a minimum inter-frame duration called Bandwidth Allocation Gap (BAG), which range from 1 ms to 128 ms, and by a maximum and minimum frame length, which are limited to Ethernet standard frame sizes. The existing AFDX architecture employs the Strict-Priority/First-In-First-Out scheduling at switch output ports and the full-duplex links with 100 Mbits/sec service capacity. An industrial configuration of AFDX network is presented later in this thesis, in Chapter 5.

1.4 Determinism and bounded delays in the network

As described earlier, in a full-duplex switched Ethernet the problem of indeterminism is shifted to the switch output port level. Since a frame arriving at an output port is buffered in a FIFO queue, it has to wait in the queue until all the frames ahead of it are transmitted and its queue is selected for transmission. As we will see in the following paragraphs, this waiting time is variable and depends directly on the instantaneous traffic in the queue. Such uncertainty of waiting time makes it very difficult to bound the

communication delay from one end of the network to the other end.

1.4.1 End-to-end delay

The end-to-end delay is the time taken by a frame to traverse the network, more precisely, it is the interval between the generation time of a frame f_i of a flow v_i at its source end-node and its arrival time at the destination end-node.

The end-to-end delay experienced by a frame can be given by the sum of two delays:

- A fixed delay introduced by technological features of the network elements; propagation delay (λ), frame transmission time (tr) and switching latency (sl).
- A variable delay introduced by waiting time (D^{wait}) in the queue of each output port traversed by the frame.

The propagation delay λ depends upon the type of transmission medium. For an Ethernet link, it is very small (usually in nanoseconds) as compared to other delays and hence it is not considered in end-to-end delay computation. Therefore for a flow v_i frame, the end-to-end delay $D_{v_i}^{ETE}$ in its path $\mathcal{P}_i$ is:

$$D_{v_i}^{ETE} = j_{v_i} + \sum_{h \in \mathcal{P}_i} (tr_{v_i} + sl^h + D_{v_i}^{wait,h})$$

where,

h is an output port in path $\mathcal{P}_i$;

j_{v_i} is the release jitter at source end-node. It is the time expressing the variability in the release of the frame with respect to its minimum inter-frame duration (see Figure 1.2). Depending on the application, it can be upper bounded. For the avionic network addressed in this thesis, it is upper bounded by 500 μsec.

tr_{v_i} is the transmission time on a link. It is fixed for a given frame length. The transmission time is upper bounded by the maximum frame length $l_{v_i}^{max}$ of flow v_i transmitted at link rate R as

$$tr_{vi} = \frac{l_{v_i}^{max}}{R}$$

sl^h is the switching latency. $sl = 0$ at end-node.
$D_{v_i}^{wait,h}$ is the waiting time in the output port queue.

The end-to-end delay experienced by a flow v_1 in a path e_x–S_x is illustrated in Figure 1.3, assuming FIFO scheduling. v_1 is assumed to be the only flow at e_x so it cannot

be delayed by any other flow frame. Thus, v_1 experiences only release jitter j_{v_1} and transmission time tr_{v_1} at e_x. At switch S_x, the forwarding process from input to the output port introduces the switching latency sl. In the given scenario, the flow v_1 arrives at S_x output port at the same time as some other competing flows $(v_2, v_3, \dots)$. Now the sequence of serving these flows is based on how they are queued in the FIFO buffer, which depends on their arrival time at S_x. In the given scenario v_1 is queued last, so it has to wait $(D_{v_1}^{wait})$ for the service of the competing flows before being transmitted. Therefore, the end-to-end delay experienced by v_1 is

$$D_{v_1}^{ETE} = j_{v_1} + tr_{v_1} + sl + D_{v_1}^{wait} + tr_{v_1}$$

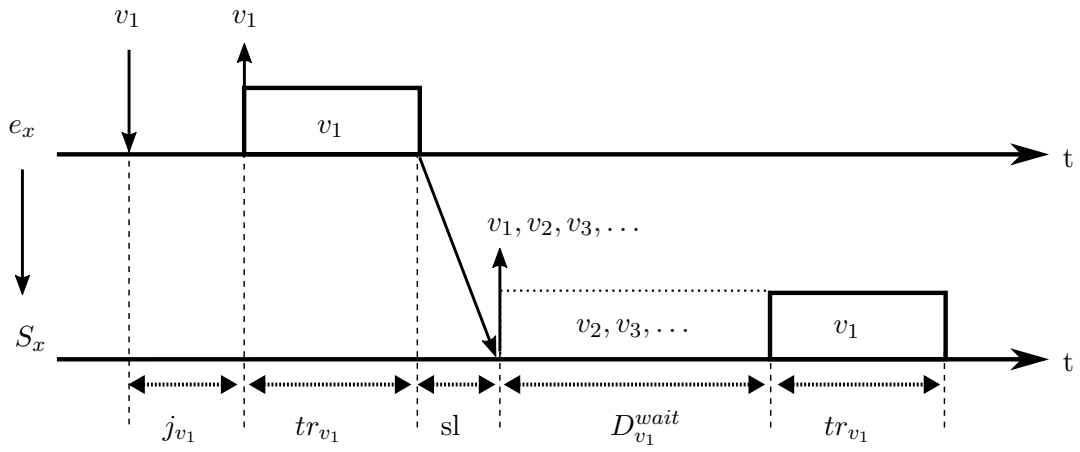


Figure 1.3 End-to-end delay illustration

Let us now have a look at the issue introduced by the uncertainty of waiting delay in the computation of exact end-to-end delay. For that purpose, let us consider the switched Ethernet network example given in Figure 1.1 and compute the end-to-end delay for v_1 in the path $\mathcal{P} = \{e_1\text{–}S_1\text{–}S_2\text{–}e_4\}$. Let us assume that each flow v_i (i= 1, 2 and 3) has a maximum frame length $l_{v_i}^{max} = 200$ bytes and minimum frame length $l_{v_i}^{min} = 100$ bytes and a minimum inter-frame arrival duration $T_{v_i} = 2$ msec. The link rate is R = 100 bits/μsec. So, the largest and smallest frames have a transmission time of 16 μsec and 8 μsec, respectively. For the simplicity of illustration, we assume that the switching latency and release jitter are negligible. Figure 1.4 shows two possible scenarios for frame transmission in the network.

In case 1, since v_1 is the only flow at e_1 it is transmitted as soon as it arrives at e_1

output port. Then, v_1 arrives at S_1 at the same time as v_2 and is queued last in the buffer. At S_2, v_1 arrives at the same time as v_3 and is queued last in the buffer. All the frames in case 1 are assumed to be of minimum length. So, the delays experienced by v_1 are: 8 μsec (transmission time) at e_1, 16 μsec (8 μsec waiting delay from v_2 + 8 μsec transmission time) at S_1 and 16 μsec (8 μsec waiting delay from v_3 + 8 μsec transmission time) at S_2. Thus, the end-to-end delay is 8+16+16 = 40 μsec.

In case 2, a similar scenario is assumed except for two modifications: first, flow v_2 is assumed to transmit a frame of maximum length but its generation instant is kept unchanged, and second, the generation time of v_3 frame is advanced by 8 μsec. In this case, since v_1 is the only flow at e_1 it experiences only transmission delay of 8 μsec. At S_1, since v_2 arrives after v_1, it cannot participate to delay v_1. So, the delay at S_1 is reduced to 8 μsec (waiting delay is zero). So, in this case, v_1 arrives earlier at S_2 as compared to case 1. However, since v_3 arrival time at S_2 is also advanced, v_1 arrives at S_2 at the same time as v_3 and is queued last in the buffer, it experiences a delay of 16 μsec (8 μsec waiting delay from v_3 + 8 μsec transmission time). The end-to-end delay, in this case, is 32 μsec.

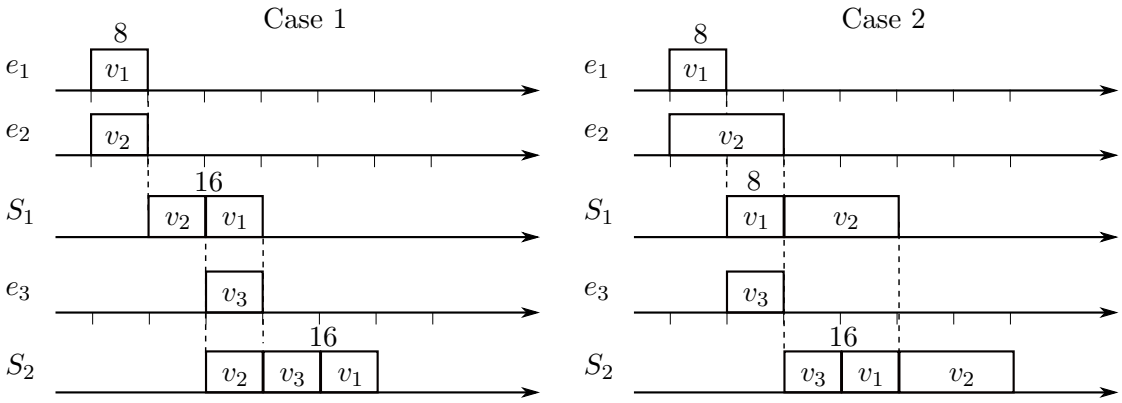


Figure 1.4 Possible frame sequences

From this example, it is clear that for a set of flows sharing a network path their relative generation time and frame sizes affect their waiting time in the buffer and eventually the end-to-end delay in the network. Indeed, since there are many possible combinations of generation time of different flow frames and these frames can be of any size (between l^{max} and l^{min}) the end-to-end delay of a flow is highly variable.

The end-to-end delay of any critical flow has to be upper-bounded. Therefore, a worst-case end-to-end delay analysis (a.k.a. Worst Case Traversal Time (WCTT) analysis) is used to provide a deterministic guarantee in the network.

1.4.2 Worst-case end-to-end delay analysis

One way to ensure the delay guarantee in the network is by adopting a deterministic approach which computes an upper limit on end-to-end delay for each flow. The knowledge of this upper limit allows ensuring that the delay for any flow in the network will always remain within the given limit. This upper limit on end-to-end delay is known as the worst-case end-to-end delay.

The deterministic approach focuses on the worst-case scenario, which corresponds to maximum congestion over the entire path of a flow in the network. However, the worst-case scenario is a rare event which is hardly ever observed on a real network. To analyse such scenarios, various approaches exist in the literature. Since this thesis considers the Ethernet network in the domain of aeronautics, we mainly focus on approaches applied to this type of networks. Main approaches are summarized in the following paragraphs.

Exhaustive analysis approach This approach performs an exhaustive analysis on all the possible scenarios of frame sequences at each output port to find the worst-case scenario. It is based on Model Checking (MC). This approach has the main advantage of computing the exact worst-case delays. MC approach [BBF+10] is based on formal verification method. It develops a model based on the system properties to performs a reachability analysis. There exist different formalisms for modelling the system. For instance, timed automata [AD94] describe the system behaviour with times. This model is composed of a set of finite automata with a set of clocks. The preliminary approach proposed in the context of an avionic switched Ethernet [CSEF06] is limited to a network configuration with up to 8 flows. This limitation is due to the fact that with the increase in the size of the network, the number of scenarios to be tested becomes very large and quickly leads toward the combinatorial explosion problem. In [ASF11, ASEF12], some properties are introduced to drastically reduce the number of scenarios to be analysed. The idea is to consider only the scenarios which are candidates to the worst-case. The resulting reduction allows the computation of exact worst-case delay for network configurations with up to 60 flows. However, up to now, this approach cannot cope with industrial size configurations (1000+ flows).

Simulation approach This approach simulates the system model based on system characteristics. Unlike the MC approach, which performs analysis on all possible scenarios, the simulation approach computes the end-to-end delays only on a set of scenarios. The accuracy of the results depends on the exactness of the system features captured by the considered model. This approach has been proposed for the temporal analysis of

a switched Ethernet network [CSEF06, SRF09, DH03, SF07, MFF07]. In the case of industrial network configuration, the number of scenarios to be analysed is very high. Thus a solution was proposed in [SF07, SRF09] which aims to focus on the part of the network configuration which has an influence on the distribution of the considered flow. [GRD02] considers a switched Ethernet network with shaped traffic. [DH03] considers a similar network with time-division. The simulation approach gives a delay distribution for a given flow. The highest delay of this distribution is known as the *observed* worst-case delay. However, this maximum observed delay is usually smaller than the exact worst-case delay, since the worst-case scenario is often a rare event which is missed by the simulation approach.

To summarise, the exhaustive approach does not scale and the simulation approach most of the time misses the worst-case. However, the worst-case end-to-end delay of critical flows has to be upper-bounded.

Several approaches have been proposed to compute such an upper-bound, namely Trajectory approach (TA) [MM05], Forward Analysis (FA) [BRBR18a], Network Calculus (NC) [BT12] etc. These approaches compute upper-bounds of the worst-case end-to-end delays by making conservative, and thus, pessimistic hypothesis.

Trajectory approach was proposed in [MM05]. TA considers the worst-case scenario that can happen to a frame along its trajectory (i.e. the sequence of nodes visited). It computes the delay upper bounds by maximizing each part of delay generated along the considered path. The principle is to concatenate all the nodes crossed by a frame in a single global node, which serves packets at the rate of the slowest node in the path. However, TA suffers an important limitation that if the global load, i.e. the sum of all the loads in the nodes from a path, is above 100% it cannot determine the upper bound of the end-to-end delay. It can therefore not be used for certification if there are some nodes with locally high loads.

The Forward Analysis was proposed in [BRBR18a, KRBR15] to overcome the drawbacks of TA while keeping tight delay bounds. In FA, all the nodes belonging to the flow path are analysed sequentially to determine the end-to-end delay. A worst-case end-to-end delay of a frame from its source node up to a given node is computed and the computation is propagated in a data-flow manner up to its destination node. Until now this approach is applicable to FIFO scheduling [BRBR18a, KRBR15] and SP/FIFO scheduling [BRBR17] based switched Ethernet network as well as to Credit Based Shaping (CBS) in Ethernet AVB network [BRBR18b].

Network Calculus approach is based on (min, +) algebra. Its principle is to model each network element as time-cumulative curves which are then used to compute the

delay incurred by a frame crossing a network element. End-to-end delay bounds are computed considering a set of local worst-case scenarios in each node.

Another approach proposed in [HL17] computes upper bound of end-to-end delays for switched networks based on some typical networking features. Unlike TA, FA and NC the approach in [HL17] does not take into account the detailed network architecture and flow paths it only considers some typically abstracted parameters like link rate, bandwidth utilisation rate, maximum flow bit rate and frame lengths. So the obtained bounds are for the whole network but not for each flow individually.

Among all the approaches mentioned above, NC approach is successfully used for the certification of avionic networks. In the rest of this thesis, we will focus on this approach.

Figure 1.5 summarizes the results provided by different approaches. The minimum delay can be trivially calculated by assuming that the waiting time is limited to scheduler latency and transmission time in each switch traversed along the path. The exact maximum delay is given by MC for small configuration. The observed worst-case delay obtained by the simulation approach varies with the duration of the simulation. For large configuration, it can be far from from the exact worst-case delay. The pessimistic assumption made by NC can (in some cases) lead to a significant difference between the upper bound and the exact worst-case delay. However, these bounds remain the only way to get a guarantee on the worst-case end-to-end delay in an industrial size network configuration. This is why we focus on the NC approach in this thesis. Let us now explain the operation of the NC approach in the following paragraphs.

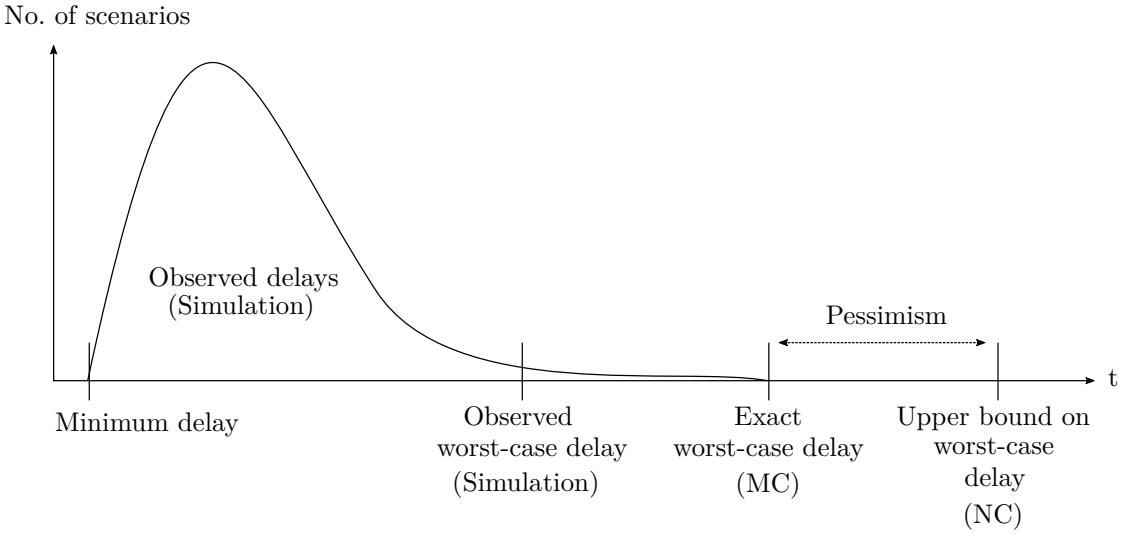


Figure 1.5 End-to-end delay results for a flow by different approaches

1.5 Deterministic method of worst-case delay analysis: Network Calculus approach

The Network Calculus (NC) is a powerful mathematical framework to analyse the performance guarantees in a network based on backlog and delay bounds. The calculations are performed through piecewise linear functions as arrival curves (a function that maps the amount of data arriving through a flow) and service curves (a function that gives the minimum amount of service provided by the server at every moment). The NC was introduced by Cruz [Cru91] for the implementation of QoS strategies on the Internet. The NC was applied to guaranteed service networks by Le Boudec et al. [Bou06, BT12]. It considers the worst-case scenario on each node visited by a flow, accounting for the maximum possible jitter introduced by the previously visited nodes. NC has been applied to switched Ethernet networks [FJJ09, CEL05, LMS05, GRD02, LH04a, LH04b, JNTW04] to guarantee real-time communication. [GRD02] presents a shared-memory architecture in each switch of a switched Ethernet and evaluates the real-time characteristics of such a network with shaped traffic using Network Calculus and computes the delay upper bounds. [LH04a] uses traffic shaping techniques to implement hard real-time distributed systems on commodity switched Ethernet and shows that the delay bounds obtained from NC depend on the traffic shaping. This work is extended in [LH04b] by using firmware offloading to lower the CPU utilization. [CEL05] focuses on the scenarios at the output port of a FIFO multiplexer. It illustrates that iteratively applying the "optimal" output bounds when flows pass through several FIFO nodes does not guarantee the overall tight bound. [LMS05] propose an improved service curve in NC to improve end-to-end delay bounds for FIFO aggregates. [FFG06] propose improvements in delay bounds by considering the serialization effect. [FJJ09] propose further improvements in delay bounds of a packet-switched network by refining the NC for the source node and the switch connected to the source node. The upper bounds computation in NC is often pessimistic, thus, the computed delay represents an upper bound on the exact worst-case delay.

In the following paragraphs, we give details on the computation implemented by NC approach.

1.5.1 NC basic concepts

In a network with the input data flow constrained by an arrival rate r, the accumulated traffic at any instant t can be given by a positive increasing function $A(t)$:

$$A(t) = \int_0^t r(x)dx$$

For such a network, NC defines an *arrival curve* (α(t)) which constraints the traffic from flow A such that:

$$A(t+s) - A(t) \leq \alpha(s)$$

which means that in an interval [t,t+s] at most $\alpha(s)$ bits of data can enter into the network.

A classical arrival curve is an affine function $\gamma_{r,b}$(t), which is defined by:

$$\gamma_{r,b}(t) = \begin{cases} rt + b & if\ t > 0 \\ 0 & otherwise \end{cases}$$

The arrival curve $\gamma_{r,b}$(t) allows a flow to emit a burst of b bits at once, but not more than r bit/s in a long-term. Such an arrival curve is shown by the red colour curve in Figure 1.6.

The service provided to flow A arriving at a network element is constrained by a service curve β(t), if and only if:

$$A^*(t) \geq \inf_{s \leq t}\{A(s) + \beta(t-s)\}$$

where, A^*(t) is the amount of traffic leaving the network element at the output, such that $A^*(t) \leq A(t)$.

A classical service curve is a rate-latency function $\beta_{R,\mathcal{T}}$(t), which is defined by

$$\beta_{R,\mathcal{T}}(t) = R[t - \mathcal{T}]^+$$

where, $[a]^+$ means $a = max\{0, a\}$.

A network element with service constrained by $\beta_{R,\mathcal{T}}$(t) delays an input flow at most by $\mathcal{T}$ time units and then offers a service rate R. Such a service curve is illustrated by the blue colour curve in Figure 1.6.

From the arrival curve α of a flow and the service curve β of a network element, NC allows to compute an upper bound of

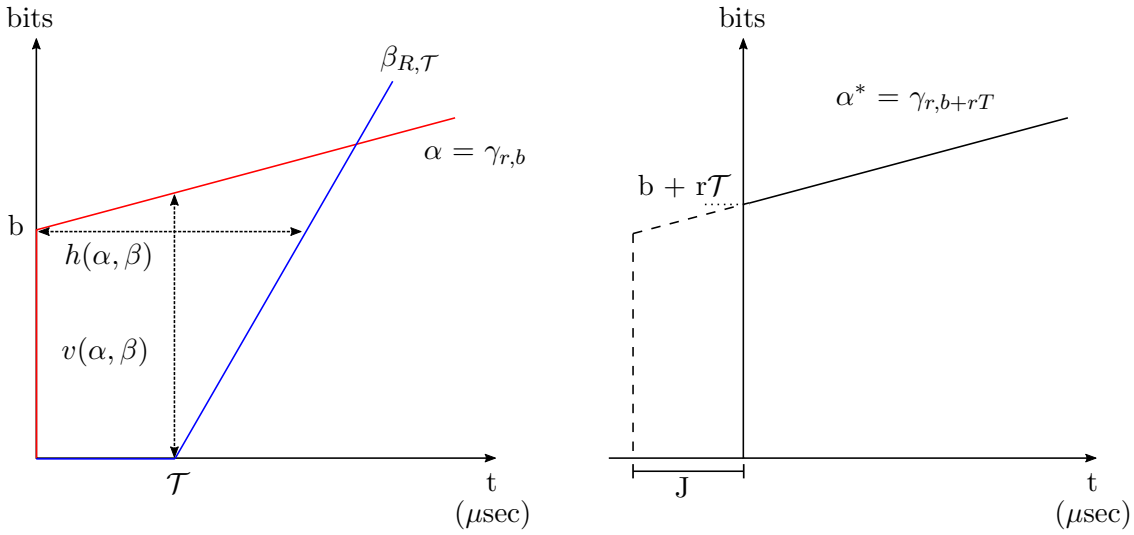


Figure 1.6 NC curves and bounds

- the backlog $x(t)$ generated by the flow at this network element,
- the delay $d(t)$ that the flow can experience at this network element,
- the flow $\alpha^*(\mathrm{t})$ at the output of this network element.

$x(t)$ and $d(t)$ are bounded by the maximum vertical and horizontal distances between α and β:

$$x(t) = v(\alpha,\beta) = \sup_{s\geq 0}\{\alpha(s) - \beta(s)\} \tag{1.1}$$

$$d(t) = h(\alpha,\beta) = \sup_{s\geq 0}(\inf\{\tau \geq 0 | \alpha(s) \leq \beta(s+\tau)\}) \tag{1.2}$$

and the output flow $\alpha^*(\mathrm{t})$ is constrained by:

$$\alpha^*(t) = \sup_{u\geq 0}\{\alpha(t+u) - \beta(u)\} \tag{1.3}$$

In a network element with service curve $\beta_{R,\mathcal{T}}(\mathrm{t})$ and a flow with arrival curve $\gamma_{r,b}(\mathrm{t})$,

if the service rate R is more than the flow arrival rate r, then:

$$v(\alpha,\beta) = b + r\mathcal{T}$$
$$h(\alpha,\beta) = \mathcal{T} + \frac{b}{R}$$
$$\alpha^*(t) = \gamma_{r,b+r\mathcal{T}} = \begin{cases} rt + (b + r\mathcal{T}) & \textit{if } t > 0 \\ 0 & \textit{otherwise} \end{cases}$$

These bounds are also illustrated in Figure 1.6. The output flow α^*(t) serves as the arrival curve of the input flow in the following network element. Therefore, it allows calculating the end-to-end delay with very low computational complexity.

1.5.2 Modelling a FIFO scheduling based real-time switched Ethernet network in NC approach

The real-time switched Ethernet network architecture (presented earlier in Section 1.3) ensures controlled input traffic (shaping) by enforcing the minimum inter-frame arrival time and the maximum frame size constraints. Under such constraints, the NC approach can be used to model the switched Ethernet network [FJJ09, CEL05, LMS05, GRD02, LH04a, LH04b, JNTW04].

Each flow traffic is constrained by a maximum frame length l^{max} and a minimum inter-frame arrival time T. Thus this flow is bounded by an arrival curve:

$$\alpha(t) = \frac{l^{max}}{T}t + l^{max}, \forall\ t > 0 \tag{1.4}$$

The source end-node is the emitter of a flow bounded by α(t) which is considered as the incoming data stream in the network. A source end-node can be a source of more than one flow. The different flows go through a FIFO multiplexer before being broadcasted on the link connecting the end-node with the first switch in the network. This FIFO multiplexer transmits the flow frames at rate R. It is represented by a service curve $\beta(t) = R[t]^+$. The destination end-node is simply modelled by a buffer accumulating the received flow frames, it is not considered as a part of the modelled network.

The Ethernet links interconnecting the network elements introduce negligible propagation delay (in nanoseconds) on the flows, thus, it is modelled as an idle transmission link with no temporal effect on the flows.

The switch forwards the incoming frame from any of multiple input ports to the specific output port that will then send the frame toward its intended destination at

rate R. A frame experiences multiple delays in forwarding process (technological delay, policing, ...). These delays are bounded by a maximum value sl. The complete switching process from the entry till the exit of a frame at the switch is modelled by a single output port. The service provided at this output port is bounded by a service curve:

$$\beta(t) = R[t - sl]^+$$

The worst-case delay of a flow in any output port (switch or end-node) is determined by the maximum horizontal distance $h(\alpha, \beta)$ between its arrival curve and the service curve provided by this output port. If there are more than one flow arriving at this output port, the arrival curve must represent the overall traffic of these flows. The simplest overall arrival curve α_o(t) is the sum of individual arrival curves of each flow arriving at the output port.

A path of a flow is modelled as the concatenation of a source node output port and several switch output ports. The Network Calculus computation starts from the source node output port along the path until the last visited switch output port. Therefore the end-to-end delay experienced by a frame following the path is the sum of delays encountered at each crossed output port.

Let us illustrate this approach by computing the upper bound on end-to-end delay for flow v_1 in the network example shown in Figure 1.1. In the path $\mathcal{P} = \{e_1\text{–}S_1\text{–}S_2\text{–}e_4\}$, the end-to-end delay upper bound for v_1 is given by NC as the sum of 3 delays, as

$$\begin{aligned} D_{v_1}^{ETE} &= d_{v_1}^{e_1}(t) + d_{v_1}^{S_1}(t) + d_{v_1}^{S_2}(t) \\ &= h(\alpha_{v_1}^{e_1}, \beta^{e_1}) + h(\alpha_o^{S_1}, \beta^{S_1}) + h(\alpha_o^{S_2}, \beta^{S_2}) \end{aligned}$$

At e_1, the arrival curve $\alpha_{v_1}^{e_1}$ of v_1 constrained by maximum frame length $l_{v_1} = 200$ bytes and an inter-frame arrival time $T_{v_1} = 2000$ μsec is

$$\alpha_{v_1}^{e_1} = \frac{(200 \times 8)}{2000}\ t + (200 \times 8) = 0.8\ t + 1600$$

Since the service rate at e_1 is 100 bits/μsec and there is no switching latency at end-node, we get service curve as

$$\beta^{e_1} = 100[t]^+$$

Thus, the delay upper bound for v_1 at e_1, from equation (1.2), is

$$d_{v_1}^{e_1}(t) = h(\alpha_{v_1}^{e_1}, \beta^{e_1}) = 16\ \mu sec$$

At S_1, the total traffic is due to the accumulation of v_1 and v_2, so the overall arrival curve $\alpha_o^{S_1}$ is the sum of arrival curves these two flows.

$$\alpha_o^{S_1} = \alpha_{v_1}^{S_1} + \alpha_{v_2}^{S_1}$$

However, the arrival of the flow frames at a switch output port can be affected by a jitter. This jitter integration in NC approach is presented in the following paragraph.

Integration of jitter

The traffic shaping at an end-node constrains the flow frames by a minimum inter-frame arrival time T. But, as the flow propagates through the network, the minimum distance between consecutive frames of this flow can vary. This is due to the fact that each frame can experience different delays at each output port. This variation of delay is known as *jitter*. The maximum jitter is defined as the difference between the worst-case delay and the best-case delay (minimum possible delay) experienced by a given flow frame before its arrival at the given output port along its path in the network [Gri04].

The maximum delay at an output port is the upper bound $d(t) = h(\alpha, \beta)$ computed by NC. The minimum delay at a port is observed when a given frame has no waiting delay in the buffer. Thus, it is the sum of switching delay sl and the transmission time at rate R. Therefore, the maximum jitter J experienced by a flow frame of size l^{max} before its arrival at n^{th} output port is:

$$J^n = \sum_{i=1}^{n-1} \left(\underbrace{d^i(t)}_{max\ delay} - \underbrace{\left(sl + \frac{l^{max}}{R} \right)}_{min\ delay} \right)$$

The integration of jitter in NC approach is provided in [Gri04], which affects the bound on output flow. This bound on an output flow at $(n-1)^{th}$ output node is also shown in Figure 1.7 and is given by:

$$\alpha^*(t) = \alpha(t + J^n)$$

Therefore, at S_1 the overall arrival curve is

$$\alpha_o^{S_1} = \alpha_{v_1}^{e_1}(t + J_{v_1}^{S_1}) + \alpha_{v_2}^{e_2}(t + J_{v_2}^{S_1})$$

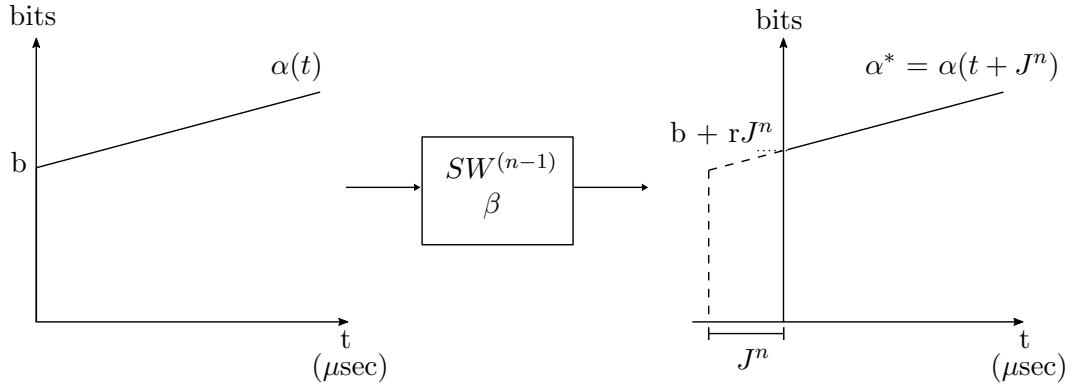


Figure 1.7 Jitter integration in arrival curve

where,

$$J_{v_1}^{S_1} = d_{v_1}^{e_1} - \left(sl^{e_1} + \frac{l_{v_1}^{max}}{R} \right) = 16 - (0 + 16) = 0$$

similarly $J_{v_2}^{S_1} = 0$, we get

$$\begin{aligned} \alpha_o^{S_1} &= \alpha_{v_1}^{e_1}(t + 0) + \alpha_{v_2}^{e_2}(t + 0) \\ &= 0.8\ t + 1600 + 0.8\ t + 1600 \\ &= 1.6\ t + 3200 \end{aligned}$$

The service provided to these cumulative flows at S_1 is constrained by the link rate 100 bits/μsec and a switching latency (let $sl = 8$ μsec). So, the service curve at S_1 is

$$\beta^{S_1} = 100[t - 8]^+$$

Thus, the delay upper bound for v_1 at S_1, from equation (1.2), is

$$d_{v_1}^{S_1}(t) = h(\alpha_o^{S_1}, \beta^{S_1}) = 40\ \mu sec$$

At S_2, the total traffic is due to the accumulation of v_1, v_2 and v_3 from two input links. However, v_1 and v_2 arrive at S_2 from the same input S_1 and thus they are serialized. The serialisation effect integration in NC approach is described in the following paragraph.

Integration of serialization

In a network element, frames arriving from different input ports are buffered in the same output port queue (FIFO). These frames are then emitted one by one on the link towards the next network element, thus, the frames are serialized. In the following output ports,

these frames remain serialized and cannot arrive at the same time. This constraint was first considered in [Gri04]. The serialisation effect results in reducing the burst of a flow by limiting it to the link rate as shown in Figure 1.8.

The serialization is integrated in the NC by considering that the arrival curve from one input port has a burst tolerance of not more than the largest frame size on the link and a rate not higher than the transmission rate of the link.

If $a \wedge b = \min(a,b)$, then the arrival curve of serialized flows (with maximum frame length l^{max}) from an input port with link rate R is given by $\alpha_s(t) = \gamma_{R,l^{max}} \wedge \alpha_o(t)$.

On a network element with n inputs, if each input link provides a serialized arrival curve of m flows from the output of previous network element, then, the overall arrival curve of all flows arriving at a network element is the sum of these serialized flows:

$$\alpha_o(t) = \sum_{i=1}^{n} \alpha_s^i(t)$$

where,

$$\alpha_s^i(t) = \gamma_{R,l^{max}} \wedge \sum_{j=1}^{m} \gamma_{r_j,l_j^{max}}$$

The use of such an arrival curve makes it possible to reduce the maximum horizontal distance from the service curve of the network element, i.e. the end-to-end delay upper bound is reduced. This optimization has allowed significant gains in the upper bounding of the delays in the switched Ethernet networks and it was used in the certification of the AFDX network of the Airbus A380.

In our example, at S_2, the frames of v_1 and v_2 are transmitted by the same input link from S_1 and they are serialized. Hence, these frames cannot arrive at the same time in the output port of S_2. So, their cumulative arrival curve (illustrated in Figure 1.8) is given as

$$\begin{aligned}
\alpha_s^{S_1}(t) &= (\gamma_{R,l_{S_1}^{max}} \wedge (\alpha_{v_1}^{S_2} + \alpha_{v_2}^{S_2})) \\
&= (\gamma_{R,l_{S_1}^{max}} \wedge (\alpha_{v_1}^{S_1}(t + J_{v_1}^{S_2}) + \alpha_{v_2}^{S_1}(t + J_{v_2}^{S_2}))) \\
&= (\gamma_{R,l_{S_1}^{max}} \wedge ((r_{v_1}t + b_{v_1}) + (r_{v_2}t + b_{v_2})))
\end{aligned}$$

Therefore, the overall arrival curve at S_2 is

$$\begin{aligned}
\alpha_o^{S_2} &= \alpha_s^{S_1}(t) + \alpha_{v_3}^{S_2} \\
&= (\gamma_{R,l_{S_1}^{max}} \wedge (\alpha_{v_1}^{S_1}(t + J_{v_1}^{S_2}) + \alpha_{v_2}^{S_1}(t + J_{v_2}^{S_2}))) + \alpha_{v_3}^{e_3}(t + J_{v_3}^{S_2})
\end{aligned}$$

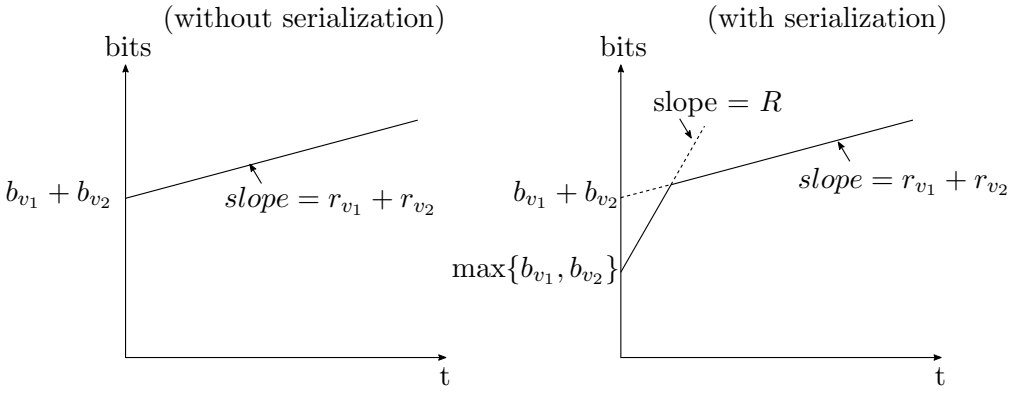


Figure 1.8 Illustration of serialisation effect in v_1 and v_2 at S_2

where,

$$J_{v_1}^{S_2} = \left(d_{v_1}^{e_1} - \left(sl^{e_1} + \frac{l_{v_1}^{max}}{R}\right)\right) + \left(d_{v_1}^{S_1} - \left(sl^{S_1} + \frac{l_{v_1}^{max}}{R}\right)\right)$$
$$= 0 + (40 - (8 + 16)) = 16\ \mu sec$$

similarly $J_{v_2}^{S_2} = 16$ μsec and $J_{v_3}^{S_2} = 0$, hence,

$$\begin{aligned}
\alpha_o^{S_2} &= (\gamma_{R,l_{S_1}^{max}} \wedge (\alpha_{v_1}^{S_1}(t+16) + \alpha_{v_2}^{S_1}(t+16))) + \alpha_{v_3}^{e_3}(t+0) \\
&= (\gamma_{R,l_{S_1}^{max}} \wedge ((0.8(t+16) + 1600) + (0.8(t+16) + 1600))) + (0.8\ t + 1600) \\
&= (\gamma_{R,l_{S_1}^{max}} \wedge ((0.8\ t + 1612.8) + (0.8\ t + 1612.8))) + (0.8\ t + 1600) \\
&[\ so,\ b_{v_1} = 1612.8\ bits\ and\ b_{v_2} = 1612.8\ bits\ ] \\
&= (\gamma_{100,1612.8} \wedge ((0.8\ t + 1612.8) + (0.8\ t + 1612.8))) + (0.8\ t + 1600) \\
&[\ since,\ R = 100\ bits/\mu sec\ and\ l_{S_1}^{max} = max\{b_{v_1}, b_{v_2}\} = 1612.8\ bits\ ] \\
&= (\gamma_{100,1612.8} \wedge (1.6\ t + 3225.6)) + (0.8\ t + 1600)
\end{aligned}$$

Since the service curve S_2 is $\beta^{S_2} = 100[t-8]^+$, the delay upper bound for v_1 at S_2, from equation (1.2), is

$$d_{v_1}^{S_2}(t) = h(\alpha_o^{S_2}, \beta^{S_2}) = 40.25\ \mu sec$$

Therefore the end-to-end delay upper bound for v_1 is

$$D_{v_1}^{ETE} = 16 + 40 + 40.25 = 96.25\ \mu sec$$

1.6 Impact of deterministic approaches in dimensioning a network

The NC approach to upper bound the worst-case delay presented in the previous section can be easily scaled to industrial size configuration. For instance, the NC has a success story of being used as a tool for certification of AFDX network in Airbus A380 aircraft. It was used to determine the upper bound of the transmission delays of all (1000+) flows in the AFDX network. These analytic bounds characterize the worst-case behaviour of traffic flows and allow dimensioning the network. The NC approach can also be used in the planning phase of the network to dimension the buffers of Ethernet switches, thanks to its capacity to compute the backlog bounds (it represents the upper limit on the congestion in the network).

However, a network design based on delay and backlog bounds provided by Network Calculus can have certain consequences in terms of inefficient resource utilisation. This is due to the fact that these bounds characterise the worst-case scenario in the network and such a scenario is a very rare event which is hardly observed in an actual implementation of the network. In order for the worst-case to occur, a data flow should experience maximum congestion at each node in its path. Such a situation is very unlikely to happen. This means a network designed to handle such scenarios is most of the time underutilised. Moreover, in some safety critical real-time systems, for instance in AFDX network, the segregation of data flows relies on the reservation of bandwidth at communication channels and, usually, the data transmitted over these channels is too few to completely utilise its reserved bandwidth, which means, most of the time the network is lightly loaded. It has been shown in the thesis of Henri Bauer [Bau11] that in an industrial AFDX network configuration of type A380 aircraft, the load on these channels is up to 33% and only 10% of the total number of channels have a load greater than 20% and for more than half of the channels the load is less than 5%. Thus, this network is almost always underused. Another fact is linked to the pessimism of the bounds computed by the NC approach. Since the delay bounds can be pessimistic, the actual traffic in case of worst-case scenario may never be high enough to utilise the maximum network capacity. And the switch buffer dimension based on pessimistic backlog bounds can lead to an expensive and over-dimensioned network architecture which further increase the effect of underutilisation of the network.

1.7 Solution to improve network resource utilisation: a QoS-aware switched Ethernet

The AFDX network is underused since it serves a quantity of data flows which is less than what it was designed to serve. In safety critical real-time applications, such a situation is common as the network is designed to provide deterministic service even in worst-case scenarios which occur rarely and most of the time the network is lightly loaded.

A solution envisioned for modern aircraft network is to use the available bandwidth in AFDX network by allowing traffic from the functions which are currently served by other dedicated networks. Since the AFDX network is currently used to serve only highly critical avionic data which has very limited traffic and the network is lightly loaded, so, the manufacturers plan to increase the load by additional non-avionic flows of other criticality levels (like inter-cabin audio communication, smoke detection, parking video, best-effort ...). Indeed, these flows can have different jitter, latency and bandwidth constraints. Thus, the main challenge with such solution is to provide fair service to additional data of different criticality without degrading the service provided to data of the highest criticality (i.e. avionic flows).

Similar situations occur in other embedded contexts. For instance, a switched Ethernet can be used as a backbone. In that case, flows from different domains are transmitted on this network, with different criticalities.

Traditional switches with FIFO scheduling do not offer any mechanism to differentiate between flows of different criticality. Consequently, the addition of new flows can have a significant impact on the delays experienced by the existing flows. Indeed with FIFO scheduling all the frames share the same queue and they are transmitted in the order of arrival, without taking into account their criticality. This phenomenon can also be shown by a simple example.

We have seen earlier (in Figure 1.4), in a switch output port controlled by a FIFO scheduler, the delay experienced by a given flow frame arriving at the output port depends upon the instantaneous traffic from other flow sharing the output port buffer. Let us again consider the scenario of case 1 in Figure 1.4 where the end-to-end delay experienced by flow v_1 is 8 μsec (transmission time at source end-node e_1) + 16 μsec (waiting delay caused by v_2 plus the transmission time at S_1) + 16 μsec (waiting delay caused by v_3 plus the transmission time at S_2) = 40 μsec. Let us now assume 5 additional flows $v_4 \ldots v_8$ with a frame length of 100 bytes (transmission time = 8 μsec) each arriving at the output port of S_2. In this case, Figure 1.9 shows a possible scenario where the frames from these additional flows can be queued in S_2 output port buffer before v_1. So these 5

frames also participate to delay v_1 which increases the total delay at S_2 to 56 μsec (48 μsec waiting delay from $v_3 \ldots v_8$ + 8 μsec transmission time) and the end-to-end delay is now increased to 80 μsec.

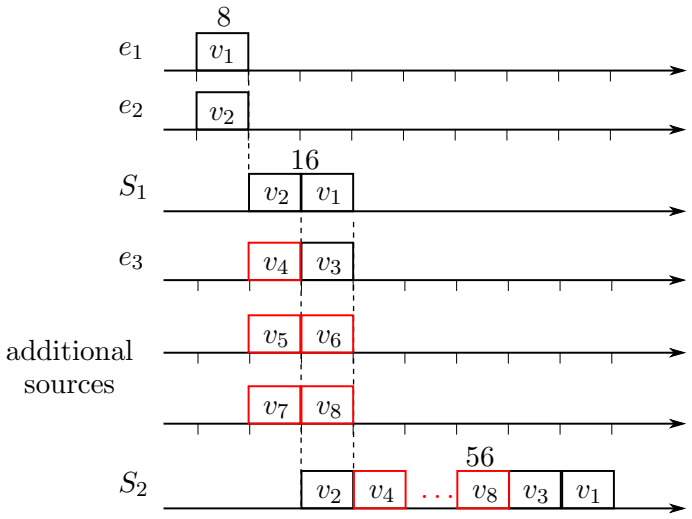


Figure 1.9 Effect of additional traffic in FIFO scheduling

The problem with such a mechanism is that we may not be able to control the amount of impact of additional flows. Indeed, the flows with high delay deadlines can accept a significant amount of increased delay caused by the addition of new flows but the flows with smaller deadlines can not accept such an impact. Therefore in this context, a simple FIFO scheduling based network architecture is not at all efficient and it becomes necessary to implement a Quality-of-Service (QoS) mechanism. QoS is a traffic prioritisation and/or resource reservation control mechanism that provides the ability to guarantee a certain level of performance to a data flow. Such a mechanism can reduce or, in some cases, avoid the impact of additional flows on the existing critical flows. Main candidate QoS mechanisms for real-time applications are presented in the following paragraphs.

Static priority queuing (SPQ) An SPQ scheduler differentiates flow frames based on a predefined priority. It manages two or more queues, where each queue is dedicated to a priority level as shown in Figure 1.10. These queues are served by the scheduler sequentially in the decreasing order of priority. For instance, first, the highest priority frames are served until their queue is empty, then, the next lower priority frames are served until their queue is empty and so on. During the service of the frames in a lower priority queue, if a new frame arrives in higher priority queue the scheduler first completes the ongoing service of the previously selected lower priority frame and then immediately

moves to the service of newly arrived higher priority frame. This way the SPQ scheduling limits the impact of the lower priority flows on the high priority flows. The only impact of a low priority flow on a high priority flow is the non-preemption delay, which is limited to the transmission time of the largest frame size of the low priority flow. So the maximum service is always guaranteed to high priority flows.

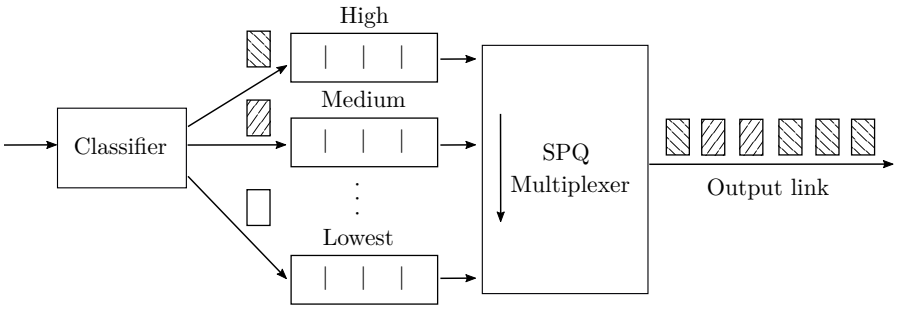


Figure 1.10 A switch output port with SPQ scheduler

On the other hand, SPQ scheduling is very unfair for the lower priority flows. A frame in a low priority queue should wait for service until all the higher priority flows are served which leads to large delays in low priority flows and even starvation. Hence, in order to preserve the maximum service for high priority flows the SPQ scheduler degrades the service of low priority flows.

The degradation in service of low priority flows is significant when the number of priority level increases and/or when there is a burst of frames with high priority. Indeed, it is necessary to define at least as many priority levels as there are flow types to differentiate. Increasing the number of priority levels results in a very limited service available to the lowest priority flows. In the extreme case, the lowest priority flows suffer starvation. The eventual consequence of starvation is a frame loss as it fails to be delivered in an acceptable time.

Thus, in a network where flows of different criticality are to be served and these flows can have different delay constraints, the SPQ scheduling fails to provide a reliable service to the flows assigned to low priority as they do not receive a fair share of network service.

One solution to mitigate this problem is shaping the highest priority traffic. Such a shaping mechanism has been introduced in Ethernet Ethernet Audio/Video Bridge (AVB) [LHWC12], through Credit Based Shaper (CBS). The goal of CBS is to ensure the provision of the maximum required bandwidth for transmission of the high priority traffic over a time sequence, without a noticeable interruption of the low priority traffic that is simultaneously transmitted. In order to achieve this, the CBS assigns a credit to the high priority flows with reserved bandwidth. The working of CBS is shown in Figure 1.11.

The initial credit is 0. As long as the credit is positive, frames in high priority queue can be transmitted. With each transmission, the credit decreases, until it eventually reaches a negative value. While the credit is in the negative range, high priority frames may no longer be transmitted. Accordingly, at this time the frames waiting in low priority queue can be processed. During the service of low priority frames and/or if there are some frames waiting in high priority queue the credit increases. As a result, the delayed high priority frames can then be transmitted back to back, following the transmission of the low priority frames. This prevents additional delays in the transmission of time-critical frames. However, this principle of CBS is non work-conserving and thus do not allow to efficiently utilise the available bandwidth.

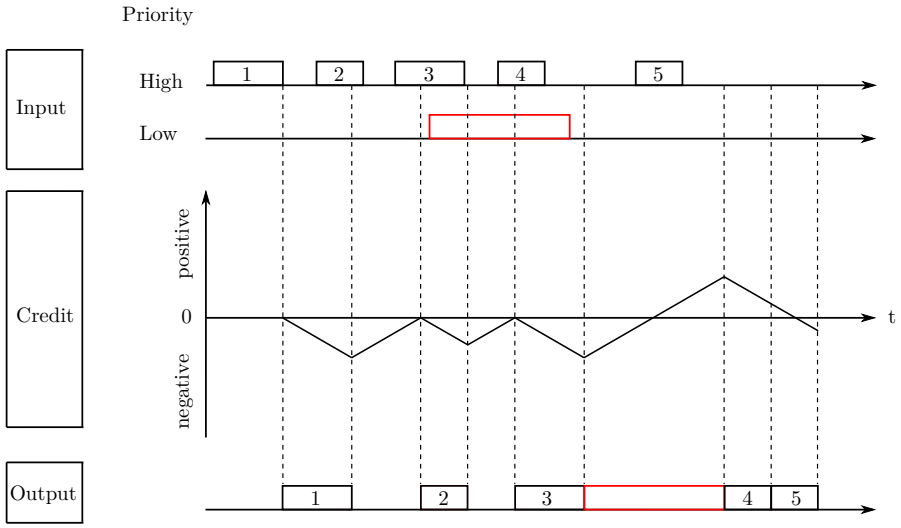


Figure 1.11 Credit based shaping of high priority flow

Another solution is allocating dedicated bandwidth to each flow class through a round robin mechanism, which allows a fair sharing of network resources in a controlled manner.

Round Robin (RR) A round robin scheduler differentiates flows based on predefined flow classes. It aims to provide a fair amount of service to these flows by reserving a minimum amount of bandwidth for each class. This minimum bandwidth is guaranteed regardless of the amount of traffic in the competing classes. The main motivation of a round robin service discipline is to provide adequate congestion control even in the presence of ill-behaved sources (i.e. the source nodes which does not implement traffic shaping). This is achieved by allocating bandwidth and buffer space in a manner which automatically ensures that ill-behaved sources can get no more than their fair share.

In a switch output port controlled by round robin scheduler, a separate FIFO queue

is managed per class as shown in Figure 1.12. The round robin algorithm allows every non-empty class queue to take turns in transferring frames on a shared channel in an infinitely repeated order. The round robin scheduling is work-conserving which means if one class queue is out of frames then the next class will be immediately served. Thus, the scheduling tries to prevent link resources from going unused.

The distribution of flows into classes is predefined and can depend on flow features (like delay constraints, throughput requirements and criticality etc) and/or network objectives (like bandwidth allocation, improve throughput, prioritize flows etc). So, one way to distribute these flows in classes can be based on the range of their deadlines and it is quite relevant to industrial applications. For example, the avionic flows have hard real-time constraints which are very small whereas other critical flows (like audio communication between cabin crew) can tolerate higher delays and other best-effort flows (like system surveillance and maintenance) can have flexible delay constraints. So, a possible solution is to have (n-1) classes of critical flows with a varying range of delay constraints and one class for best-effort flows.

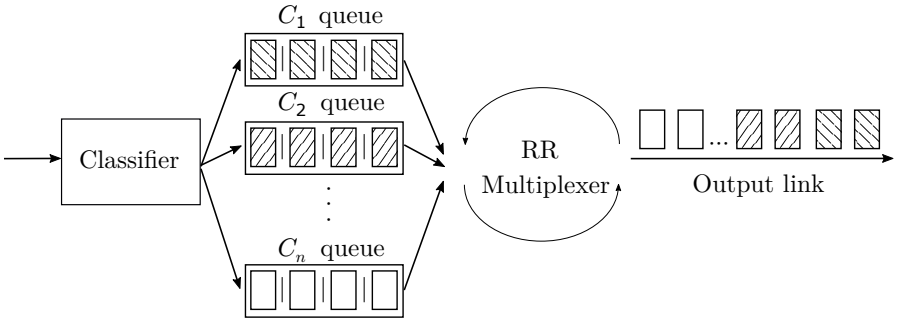


Figure 1.12 A switch output port with Round Robin scheduler

A round robin scheduler provides service according to the *credit* allocated to each class. The definition of credit depends upon the type of round-robin algorithm. The round-robin algorithms are classified into two major categories based on their credit assignment mechanism. The first category assigns credit in terms of *number of frames* so that each class is allowed to transmit at most the assigned number of frames in each round. The second category assigns credit in terms of *number of bits or bytes* which can be transferred from each class queue in each round. The Weighted Round Robin and Deficit Round Robin scheduling are practical examples of these two categories which are gaining high interest in industrial application as they can provide fair sharing of bandwidth with O(1) complexity of implementation.

Weighted Round Robin (WRR) A WRR scheduler allows a certain number of frame transmission per class per round robin cycle. Each class is assigned a *weight* that represents the number of frames that a class can transmit in each round. These classes are then served one by one in each round. Thus, each class is allowed to transmit some frames at least once in a given round.

Let us have a look at the operation of WRR scheduling on a small example of a switch output port with 2 classes C_1 and C_2 shown in Figure 1.13. Each class is assumed to have 10 frames queued in its buffer. Each frame in C_1 is of 100 bytes and that in C_2 is of 50 bytes. Let us assume a credit of 2 frames for each class. In each round, the scheduler selects each class sequentially. Each time a frame is transmitted from the selected class its credit is reduced by one frame. Initially, C_1 is selected. Since C_1 has 10 frames waiting in buffer and it has a credit of 2 frames, it is allowed to transmit first 2 frames queued in the buffer leading to a transmission of $2 \times 100 = 200$ bytes. Now the C_1 credit is reduced to 0, so, the remaining 8 frames should wait for the next round. Now class C_2 is selected and assigned a credit of 2 frames. The credit of C_2 is reduced to 0 with the transmission of 2 frames (total bytes transmitted $= 2 \times 50 = 100$ bytes). The first round ends with the service of C_2. In the next round, the same process is repeated for each class.

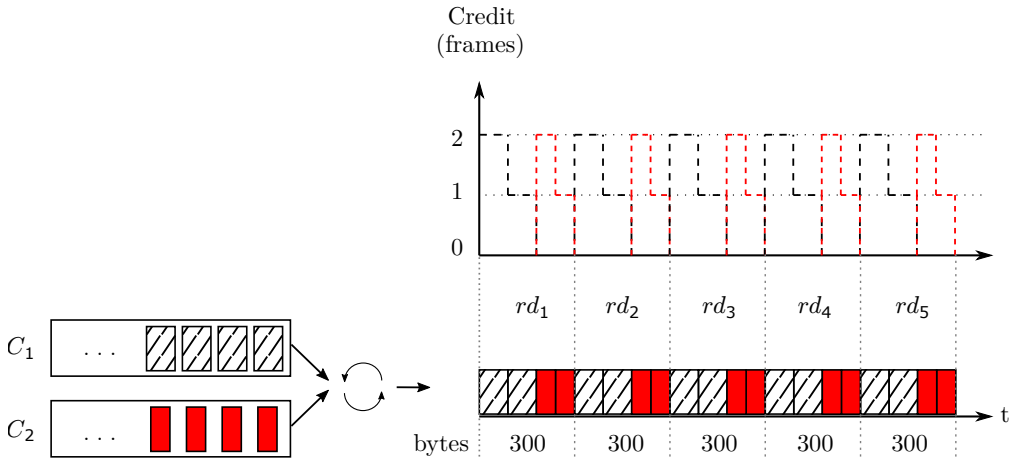


Figure 1.13 WRR scheduling

Such scheduling mechanism allows a strict control over the number of frames that each flow class transmits per round and at the same time it ensures that no class is left without service. However, WRR scheduler is limited by its inability to take frame size into consideration. When there is a big difference in the size of frames in each class, one class could transmit frames of minimum length while other classes transmit frames of maximum length, thus, eventually consuming very few bandwidth than what was

allocated to it. We will illustrate this problem along with a thorough analysis of WRR scheduling in Chapter 6.

Deficit Round Robin (DRR) Basically, DRR [SV96, PG93] is a variation of WRR which aims to achieve a better QoS by fair sharing of available network bandwidth among the flow classes even in the presence of flows with a large range of frame lengths. A DRR service is divided into rounds. Each active class is assigned a quantum by the DRR scheduler. The quantum assigned to a class is defined as the service (in number of bytes) that its flows should receive during each round robin service opportunity. During some service opportunity, a class may not be able to transmit a frame because doing so would cause this class to exceed its allocated quantum. So, the scheduler maintains a deficit count for each class, which is the difference between the amount of data actually transmitted in the given round and the amount that was assigned to this class. In the next round, these deficits are added to the quantum value of corresponding classes. Thus, a class that received very little service in a given round is offered an opportunity to receive more service in the next round.

Let us have a look at the operation of DRR scheduling on a small example of a switch output port, shown in Figure 1.14, serving 2 classes (C_1 and C_2) where 10 frames are queued in each class buffer. Each frame in C_1 is of 100 bytes and that in C_2 is of 50 bytes. Let us assume a credit of 250 bytes for each class. In each round, the scheduler selects each class sequentially. Each time a frame is transmitted from the selected class its credit is reduced by the size of this frame. Initially, C_1 is selected and assigned a credit of 250 bytes. Since C_1 has 10 frames of 100 bytes, it consumes the credit of 200 bytes in the transmission of $\frac{200}{100} = 2$ frames. C_1 credit is now reduced to 50 bytes. Since the next frame in C_1 queue is of 100 bytes it cannot be transmitted and the remaining credit of 50 bytes is stored as a deficit so that C_1 will be able to transmit 250+50 = 300 bytes in the next round. Next, C_2 is selected and assigned a credit of 250 bytes. Since each frame in C_2 is of 50 bytes, it consumes the credit of 250 bytes in the transmission of $\frac{250}{50} = 5$ frames. The first round ends with the service of C_2. In the next round, the classes are served in a similar manner.

Such scheduling mechanism allows precise control over the bandwidth available to each traffic class and makes it highly suitable to handle flows of mixed criticality with different delay constraints.

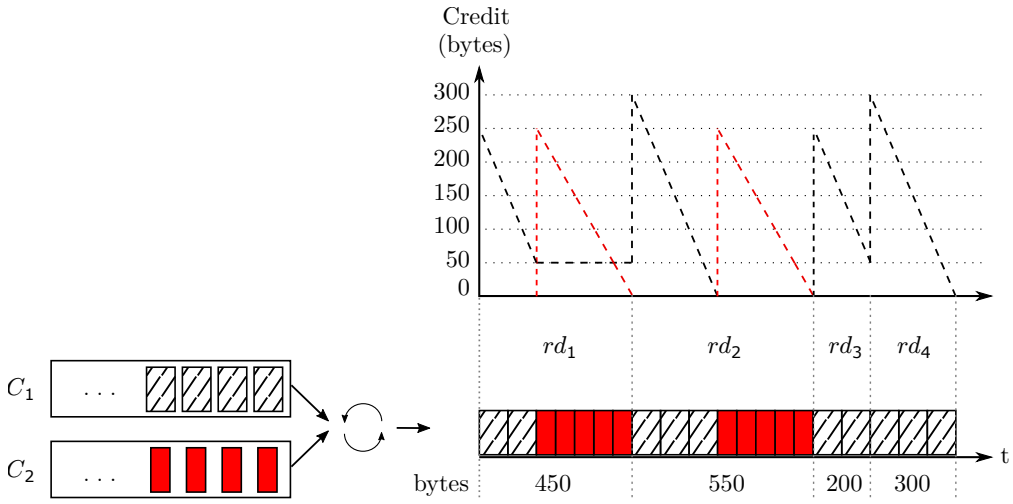


Figure 1.14 DRR scheduling

1.8 Thesis scope and contributions

The deployment of QoS mechanism in a real-time switched Ethernet network can give a possibility to efficiently serve mixed criticality flows having different delay constraints. In particular, in the absence of a global synchronisation clock, the DRR and WRR scheduling allow differentiating these flows into classes such that the impact of one flow class on another class is limited and a certain amount of minimum service can be assured for each class. Such a mechanism permits to increase the amount of flows that can be transmitted over the network and eventually improves the network utilisation rate. This is why a real-time switched Ethernet network with such QoS mechanism is of great interest for industrial applications. Furthermore, this solution is envisioned by the aircraft manufacturer Airbus for their modern aircraft deploying AFDX backbone. This is the reason why, in this thesis, we focus on the WCTT analysis of such a network.

In the shared network environment, the availability of network resources, such as bandwidth, to different flows plays an important role in the overall performance of the network since it could affect the delay experienced by these flows. In the above mentioned schedulers a certain amount of bandwidth can be assured on per class basis. Since different flows can have varying delay constraints we propose to distribute these flows into classes based on these constraints such that we can have (n-1) classes of critical flows with different delay constraint per class and 1 class for best-effort flows which may not have a strict constraint. Now the bandwidth allocated to each class can be optimised to satisfy their constraints. Actually, it comes down to the problem of optimal credit

assignment per class, which is also addressed in this thesis.

In order to provide the deterministic guarantee in this network, a method allowing to handle shared network must be used to compute the WCTT bounds. So far the Network Calculus is the only scalable and reliable approach to compute the worst-case delay bounds in such context (with round robin schedulers). The main issue with the Network Calculus is the pessimism: the computed delay bounds are often larger than the attainable one. An important contribution of this thesis concerns the optimisation of Network Calculus approach to mitigate its pessimism.

Methodology: In this chapter, we have presented the importance of deterministic approach (Network Calculus) and the need of implementing QoS mechanism in real-time switched Ethernet network with mixed criticality flows. We have selected to study the network architecture based on DRR and WRR scheduling given their fair resource sharing capacity at low implementation cost and the interest shown by the European multinational aerospace corporation Airbus.

In the rest of the thesis, we focus on WCTT analysis in this network. For that purpose, we first illustrate the Network Calculus approach, originally presented by Marc Boyer et al in [BSS12] that allows computing the delay bounds in the shared network environment. This approach is applied to DRR scheduler based switched Ethernet network.

In the next step, we show the link between the delay bounds and the credit assignment in DRR scheduler and how it can be used to optimise the resource allocation. We propose a credit assignment algorithm that allows the critical flows to receive sufficient bandwidth so that they can be served in the limit of their delay constraints and the non-critical flows receive maximum residual bandwidth which can be used to reduce their delay. We also show that the delay bound computation based on Boyer's Network Calculus approach can be pessimistic. In order to mitigate this pessimism we propose some optimisation that allows us to compute tight delay bounds.

The evaluation of our credit assignment algorithm and optimised Network Calculus approach is presented through a case study on an industrial size AFDX network configuration. Actually, we have performed different case studies: one to evaluate our proposed solutions and another to compare the performance of DRR and WRR scheduling. The results from the first case study show the significant improvements in delay bounds and overall performance of the network. The second case study reveals the advantages and limitations of each scheduling which can be used as a guideline for the QoS implementation in industrial application.

1.9 Conclusion

In this chapter, we have seen the evolution of switched Ethernet network to satisfy the needs of real-time industrial applications. The deployment of a QoS mechanism and increased traffic in this network can improve the utility of network resources while still providing deterministic service to flows having hard real-time constraints.

Providing deterministic guarantee in complex network architectures is a challenging task. This guarantee is provided in real-time switched Ethernet network by constrained data flow and by upper bounding the end-to-end delays using a deterministic computation technique. Such delay bound computation using the Network Calculus approach in a QoS enabled network architecture is presented in the next chapter.

Chapter 2

A QoS-Aware Switched Ethernet Network

Contents

2.1 Introduction 43

2.2 Real-time switched Ethernet network architecture with DRR scheduling 44

2.3 Overview of DRR scheduling 45

2.3.1 DRR working principle 45

2.3.2 Bandwidth sharing in DRR scheduler 47

2.4 Worst-Case Traversal Time analysis 50

2.4.1 DRR scheduler latency 50

2.4.2 Modelling real-time switched Ethernet network with DRR scheduling in Network Calculus 56

2.5 Conclusion 58

2.1 Introduction

The aim of this chapter is to present a real-time switched Ethernet network architecture based on DRR scheduling and to illustrate the existing NC approach for WCTT analysis.

Section 2.2 shows a DRR scheduler based real-time switched Ethernet network architecture. Section 2.3 gives the detailed description of DRR scheduling principle. The WCTT analysis is presented in Section 2.4. Section 2.5 concludes the chapter.

2.2 Real-time switched Ethernet network architecture with DRR scheduling

We consider a real-time switched Ethernet network similar to the one described in the previous chapter (Section 1.3) but with the switches upgraded to support a flow differentiation mechanism. Figure 2.1 shows a very basic example of this network where a switch S_1 interconnects 6 end-nodes (e_1 – e_6) forwarding 21 flows (v_1 – v_{21}) over full-duplex links at a link rate $R = 100$ bits/μsec. The switch output port is controlled by a DRR scheduler which differentiate flows into 3 predefined classes (C_1 – C_3) where a FIFO queues is associated to each class. The flow frames from these queues are served in rounds based on a credit assigned to the corresponding class queue.

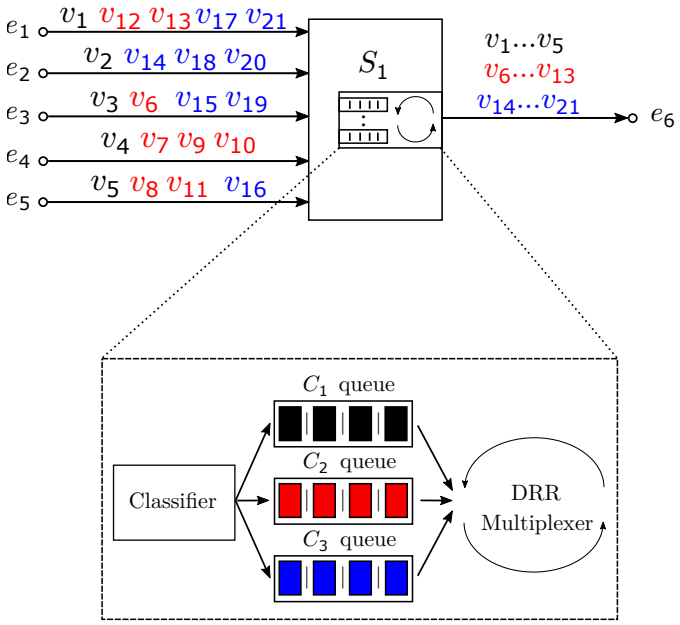


Figure 2.1 An example of a switched Ethernet network with DRR scheduling

The flow differentiation in this example is arbitrarily assumed, such that: v_1 to v_5 are classified as C_1, v_6 to v_{13} as C_2 and the remaining flows v_{14} to v_{21} as C_3. Each flow is bounded at its source end-node by a minimum inter-frame time (T_i), a maximum (l_i^{max}) frame length and a minimum (l_i^{min}) frame length (given in Table 2.1). So the maximum and minimum frame arriving in C_x buffer is also constrained to

$$l_{C_x}^{max} = \max_{i \in \mathcal{F}_{C_x}} \{l_i^{max}\} \quad and \quad l_{C_x}^{min} = \min_{i \in \mathcal{F}_{C_x}} \{l_i^{min}\} \tag{2.1}$$

where, $\mathcal{F}_{C_x}$ is the set of flows of C_x.

Table 2.1 Flow specifications

Flows	T_i (msec)	Frame lengths (bytes)
$v_6, v_{12}, v_{13}, v_{20}$	128	80–100
$v_1, v_7, v_8, v_9, v_{17}$	128	80–99
$v_2, v_4, v_5, v_{10}, v_{16}, v_{18}, v_{21}$	64	80–100
$v_3, v_{11}, v_{14}, v_{15}, v_{19}$	64	80–99

2.3 Overview of DRR scheduling

2.3.1 DRR working principle

The basic idea of DRR is to assign a quantum $Q^h_{C_x}$ to each *active* class C_x at a given switch output port h. A class is said to be active when it has at least one frame in its class buffer waiting to be transmitted. $Q^h_{C_x}$ is the number of bytes and it corresponds to the minimum credit that should be assigned to C_x in each scheduling round at h. In any round, the total credit of C_x is the sum of its quantum $Q^h_{C_x}$ and a deficit $\Delta^h_{C_x}$. The deficit $\Delta^h_{C_x}$ in any round is the unused credit of C_x from the previous round. In order to assure that all the active classes receive some service in each round the credit assigned to C_x must be such that it allows transmission of at least one frame of any size in C_x at h. Thus, $Q^h_{C_x}$ cannot be less than the maximum frame size of C_x flows. Therefore,

$$Q^h_{C_x} \geq l^{max,h}_{C_x} \tag{2.2}$$

The working principle of DRR scheduler serving n classes is shown in Algorithm 1. Initially, the deficit $\Delta^h_{C_i}$ for each class is set to 0 (lines 1–3). Then, the class queues are selected in a round robin order (lines 5–18). In each round, only active classes are served (line 6) i.e. credit is assigned only to the non-empty queues. Each active class queue receives a credit of $Q^h_{C_i} + \Delta^h_{C_i}$ in each round (line 7). For the selected class queue, the frames are sent as long as the queue is not empty (i.e. selected class remains active) and the credit is larger than the size of the head-of-line frame (lines 8–12). If the queue becomes empty, the deficit is reset to 0 (lines 13-14), otherwise, the deficit is set to the remaining credit value (lines 15–17).

Let us illustrate the operation of DRR scheduling on the network example discussed earlier (Figure 2.1). One possible scenario of DRR scheduling at S_1 output port is shown

Algorithm 1: DRR Algorithm

Input : Per class quantum: $Q^h_{C_1} \dots Q^h_{C_n}$ (Integer)
Data: Per class deficit: $\Delta^h_{C_1} \dots \Delta^h_{C_n}$ (Integer)
Data: Per class credit: $credit^h_{C_1} \dots credit^h_{C_n}$ (Integer)
Data: Counter: i (Integer)

```
1  for i = 1 to n do
2  |  Δ^h_{C_i} ← 0 ;
3  end
4  while true do
5  |  for i = 1 to n do
6  |  |  if isActive(C_i) then
7  |  |  |  credit^h_{C_i} ← Q^h_{C_i} + Δ^h_{C_i};
8  |  |  |  while (isActive (C_i)) and (size(headFrame(C_i)) ≤ credit^h_{C_i}) do
9  |  |  |  |  send(headFrame(C_i));
10 |  |  |  |  credit^h_{C_i} ← credit^h_{C_i} − size(headFrame(C_i));
11 |  |  |  |  remove(headFrame(C_i));
12 |  |  |  end
13 |  |  |  if isNotActive(C_i) then
14 |  |  |  |  Δ^h_{C_i} ← 0
15 |  |  |  else
16 |  |  |  |  Δ^h_{C_i} ← credit^h_{C_i}
17 |  |  |  end
18 |  end
19 end
```

in Figure 2.2 where each class C_x (x = 1,2 and 3) is assigned equal quantum $Q^{S_1}_{C_x}$ of 199 bytes which is larger than their biggest frame ($l^{max}_{C_x} = 100$ bytes).

The service starts at instant t_1 when at least one of the classes is active. The decision about the class that will be served first depends upon the arrival instant of the frames in the corresponding class queues. Let us assume, C_2 is selected first. Initially the deficit count Δ is zero for all classes. In first round rd_1, C_2 receives $credit_{C_2} = Q^{S_1}_{C_2} + \Delta^{S_1}_{C_2} = 199 + 0 = 199$ bytes which allow it to serve the first two frames v_6 and v_7 of total 100 + 99 = 199 bytes. Similarly, next active class C_3 also receives $credit_{C_3} = Q^{S_1}_{C_3} + \Delta^{S_1}_{C_3} = 199 + 0 = 199$ bytes which allow it to serve v_{21} and v_{14}. The next active class is C_1. It also receives $credit_{C_1} = Q^{S_1}_{C_1} + \Delta^{S_1}_{C_1} = 199$ bytes. Since the first two frames in C_1 are v_5 and v_4 of 100 + 100 = 200 bytes which are larger than available credit, C_1 can transmit only first frame v_5 of 100 bytes in this round. Thus, the unused credit of C_1 in rd_1 is $\Delta^{S_1}_{C_1} = 99$ bytes which can be assigned to it in the next round so that the total credit

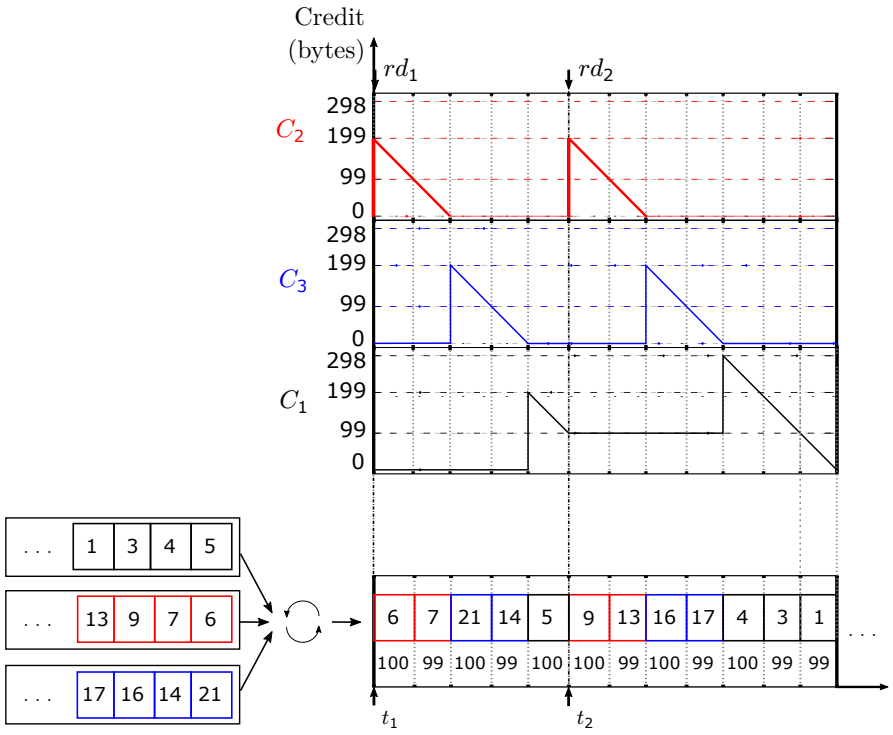


Figure 2.2 Illustration of DRR scheduling

that can be used by C_1 in rd_2 is $credit_{C_1} = 199 + 99 = 298$ bytes. In the next round the service continues in the same manner which is also shown in Figure 2.2.

2.3.2 Bandwidth sharing in DRR scheduler

Since all the active classes are served in each DRR service round, the maximum link capacity is shared among these classes over their active period. So, each class gets a fraction of the maximum available bandwidth R in the given period. As shown in Figure 2.2, the amount of data served from a class C_x queue in a given round depends upon the frame sizes and available credit. Consequently, the bandwidth utilised by this class is variable in each round. However, on the long-term, the average amount of data served per round per class is equal to the quantum assigned to this class. This is possible thanks to the capacity of DRR scheduler to take into account the leftover credit (i.e. deficit) of C_x in each round. This can also be observed in Figure 2.2 where C_1 is not able to consume its full credit (199 bytes) in the first round rd_1 leaving a deficit of 99 bytes which is utilised in the next round rd_2 allowing transmission of 298 bytes. So the average amount of bytes served from C_1 in 2 rounds is $\frac{100+298}{2} = 199$ bytes per round, which is

equal to its quantum $Q_{C_1}^{S_1}$. Similarly, C_2 and C_3 also receive the average service equal to their respective quantum values. Therefore, the service rate received by an active class C_x is defined as:

Definition 1. *Theoretical service rate ($\rho_{C_x}^h$): The theoretical service rate is the guaranteed average service rate that a traffic class C_x should get, on the long-term, even in the presence of maximum traffic from the competing classes.*

At an output port serving n traffic classes, since each class can serve on average quantum amount of data, the theoretical service rate is

$$\rho_{C_x}^h = \frac{Q_{C_x}^h}{\sum\limits_{1 \leq j \leq n} Q_{C_j}^h} \times R \tag{2.3}$$

It is worth noting that the theoretical service rate is defined on long-term. However in a given (shorter) time interval the service provided to a class might be more, less, or equal as compared to the theoretical one.

Definition 2. *Actual service rate: The actual service rate is the average service rate received by a traffic class C_x in a short time interval.*

At an output port serving n traffic classes, since the amount of data served from each class depends upon the size of frames present in class buffer and the credit available in a given round, a class C_x can receive different average service rate between two consecutive service opportunities. In other words, in an interval $[t_s, t_e]$, the average service rate of a given class C_x depends on the frames which effectively cross the output port. Therefore, the actual service rate is given by:

$$\rho_{act,C_x}^h = \frac{N_{C_x}}{\sum\limits_{j=1}^{n} N_{C_j}} \times R \tag{2.4}$$

where, N_{C_i} is the number of bytes served from C_i queue in the interval $[t_s, t_e]$ and n is the number of classes at h. For an inactive class C_j, $N_{C_j} = 0$.

Let us again consider the scenario discussed in Figure 2.2 to illustrate the difference between the theoretical service rate and actual service rate. Let $[t_{s_i,C_x}, t_{e_i,C_x}]$ be the interval between the i^{th} and $(i+1)^{th}$ service of C_x flows, then Figure 2.3 shows the service rates for C_2 flows. In the first interval $[t_{s_1,C_2}, t_{e_1,C_2}]$, C_2 consumes its full credit in the transmission of 199 bytes and then waits for its next service while C_3 and C_1 transmit

199 bytes and 100 bytes respectively. This leads to the average service rate received by C_2 in $[t_{s_1,C_2}, t_{e_1,C_2}]$, given by equation (2.4), as

$$\rho_{act,C_2} = \frac{199}{199+199+100} \times 100 = 39.95\ bits/\mu sec$$

Whereas in interval $[t_{s_2,C_2}, t_{e_2,C_2}]$, C_2 and C_3 transmit 199 bytes each and C_1 transmits

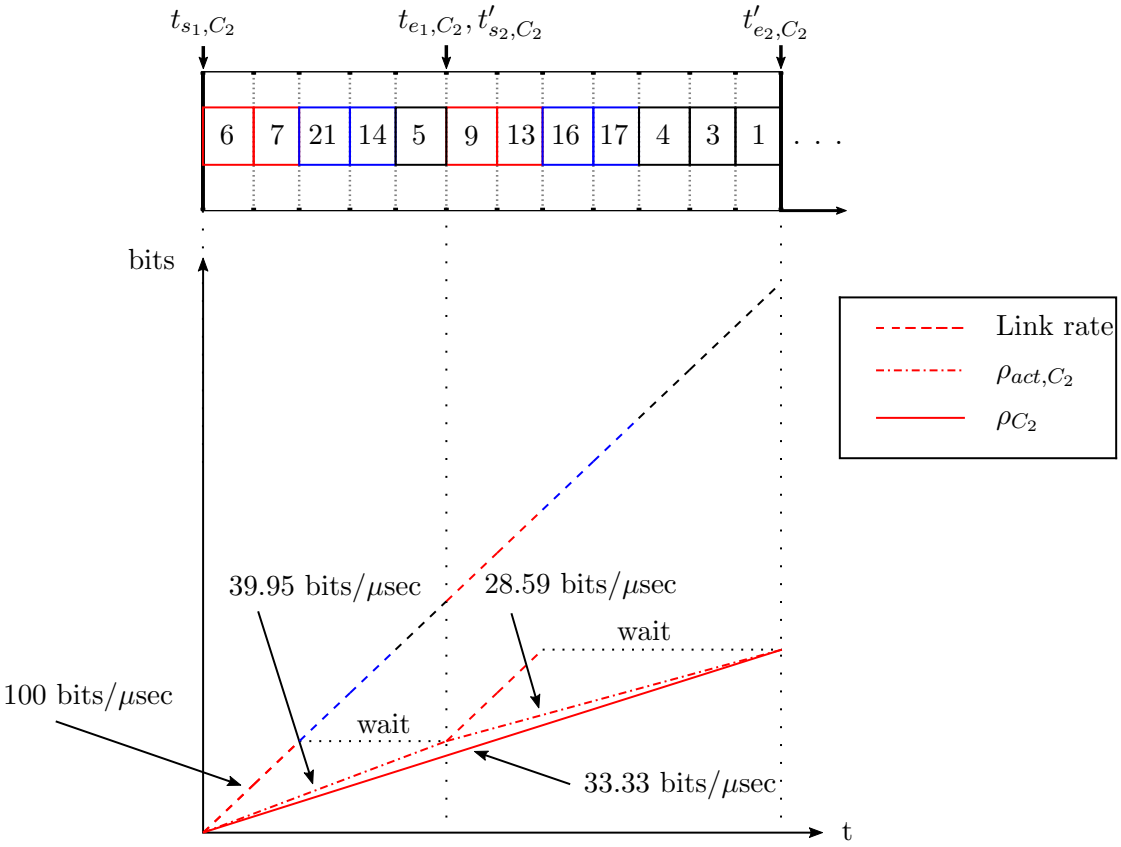


Figure 2.3 Theoretical and actual service rate in C_2

298 bytes, so the actual service rate in this interval is reduced to

$$\rho_{act,C_2} = \frac{199}{199+199+298} \times 100 = 28.59\ bits/\mu sec$$

However, despite the variation in service rate received by C_2 flows over these small intervals, the average amount of transmission from each class on a large interval (say $[t_{s_1,C_2}, t_{e_2,C_2}]$) is 199 bytes, which is equal to their quantum values, thus the long term service rate is constant. This long term service rate is lower bounded by the theoretical

service rate. For C_1, C_2 and C_3, the theoretical service rate given by equation 2.3, is

$$\rho_{C_1} = \rho_{C_2} = \rho_{C_3} = \frac{199}{199 + 199 + 199} \times 100 = 33.33\ bits/\mu sec$$

2.4 Worst-Case Traversal Time analysis

The deterministic guarantee in a network with multiplexed critical and non-critical flows can be provided by an upper bound on the end-to-end delay for each time-constrained flow in the network. This guarantee was provided by NC in the context of the network with FIFO scheduling. Likewise, the addition of new categories of traffic requires a methodology adapted to obtain similar bounds in a context of flow differentiation. The NC was extended by Boyer et al. [BSS12] to compute the end-to-end delay bounds in the network with DRR schedulers.

The aim of worst-case delay analysis is to upper bound the delay of a given flow. This delay corresponds to the minimum service received by this flow. In a DRR scheduler, service received by a class C_x flows is affected by the latency introduced by the scheduling process. The scheduler latency is one drawback of DRR scheduler and it has been evaluated in [VS96, LMS02, KS02, BSS12]. All these papers see the residual service of a flow as a constant-rate or latency-rate ($\mathcal{LR}-$) server and each paper provides its own evaluation of scheduler latency where [KS02] gives tighter bound as compared to any other evaluation. The NC model proposed by Boyer et al. considers the same scheduler latency as the one derived in [KS02].

In the following paragraphs, we present the DRR scheduler latency and then we show the NC approach for worst-case end-to-end delay analysis based on the original work from Boyer et al. [BSS12]

2.4.1 DRR scheduler latency

The DRR scheduler latency in [KS02] is based on the measure of the cumulative time that a flow has to wait until it begins receiving service at its guaranteed service rate and it is defined as follows.

Definition 3. *Scheduler latency ($\Theta^h_{C_x}$): A DRR scheduler latency is the maximum delay experienced by a traffic class C_x before it is served at its theoretical service rate $\rho^h_{C_x}$.*

According to [KS02], in DRR scheduler, a traffic class C_x can experience multiple delays at the beginning of its active period, before being served at its theoretical service rate. More precisely, it can experience two delays:

- a delay $X^h_{C_x}$ before class C_x receives service for the first time in its active period,
- a delay $Y^h_{C_x}$ to take into account the fact that, when C_x receives service for the first time, it can be a reduced service (i.e. it is served at less than its theoretical service rate).

These delays are illustrated in Figure 2.4, where:

- t_i is the starting time of round i,
- First round rd_1 starts at the arrival time of a frame of the class under study (C_1) with no backlog for this class,
- $X^h_{C_x}$ is part of the first round, starting at time t_1 and ending at time t'_1,
- $Y^h_{C_x}$ is part of the second round, starting at time t'_2 and ending at time $t''2$.

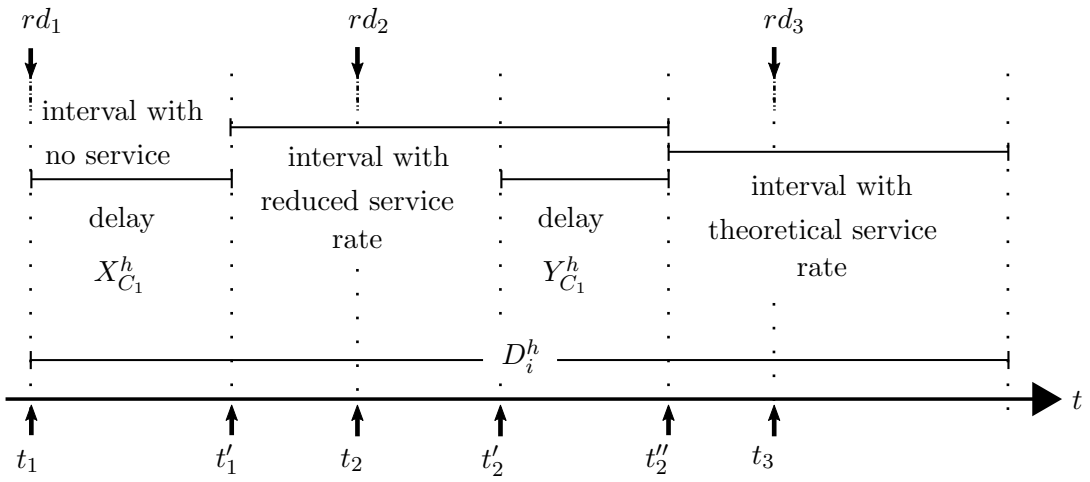


Figure 2.4 DRR scheduler latency

Let us illustrate the DRR scheduler latency by considering a possible scenario for C_1 flows (shown in Figure 2.5) at the output port of S_2 in Figure 2.1.

At t_0, 5 frames arrive: four belonging to class C_2 (from flows v_{12}, v_6, v_7 and v_8 in this order in the queue) and one belonging to class C_3 (from flow v_{20}). We assume that C_2 is served first. It receives $credit_{C_2} = Q_{C_2} = 199$ bytes and consumes 100 bytes in transmission of a frame from v_{12}. Since the next head-of-line frame (of 100 bytes from v_6) is larger than remaining $credit_{C_2} = 199 - 100 = 99$ bytes, it cannot be transmitted and the scheduler checks for the next active class (C_3). This leaves a deficit $\Delta_{C_2} = 99$ bytes. Now, C_3 receives $credit_{C_3} = Q_{C_3} = 199$ bytes and consumes 100 bytes in the

transmission of a frame from v_{20} and the credit is reduced to 99 bytes. Meanwhile, 5 new frames arrived in C_2 (from flow v_9) and C_3 (from flows v_{21}, v_{14}, v_{15} and v_{16}). The next frame in C_3 (from v_{21}) is larger than current remaining credit and cannot be transmitted, thus, this leaves a deficit $\Delta_{C_3} = 99$ bytes.

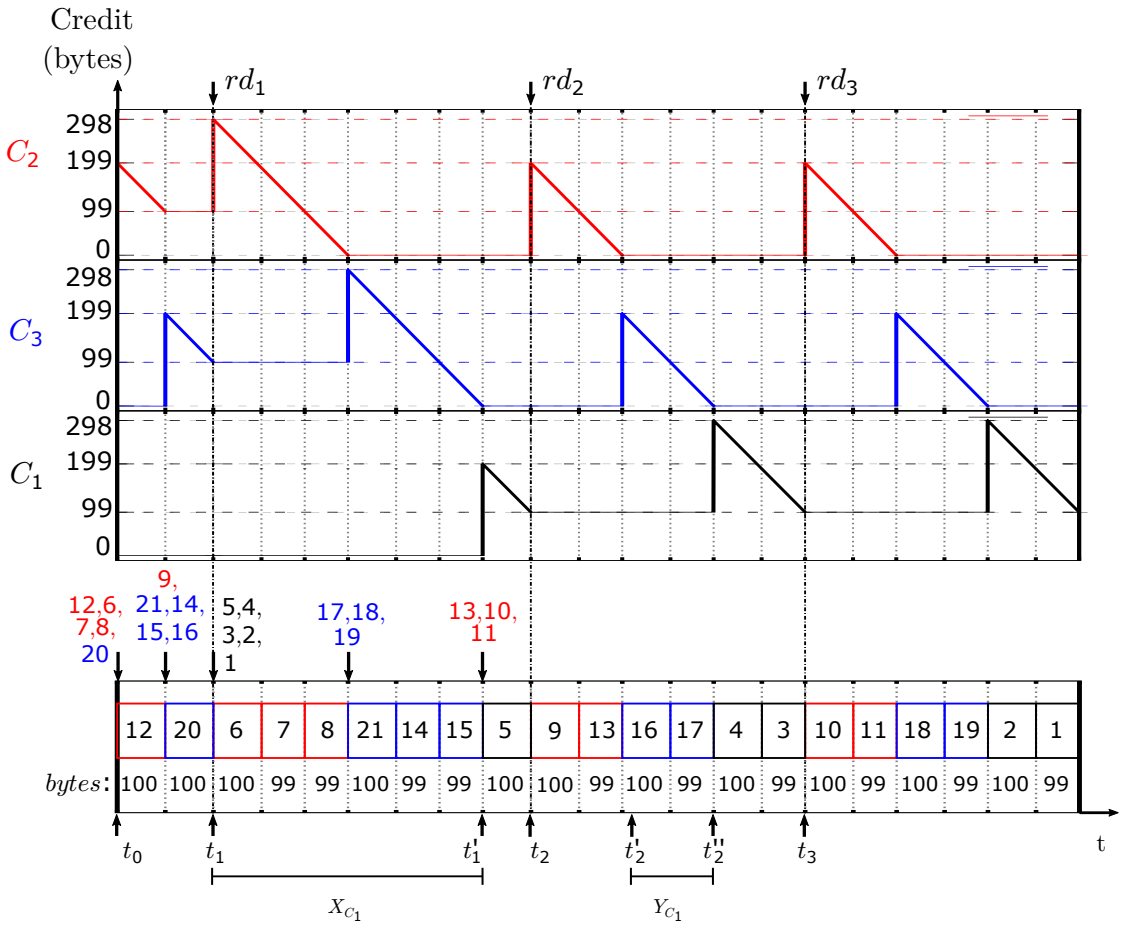


Figure 2.5 Illustration of DRR scheduler latency at S_1 output port

At t_1, C_1 becomes active with the arrival of 5 frames (from flows v_5, v_4, v_3, v_2, and v_1). However, in the worst-case for C_1 flows, the first frame arrives in C_1 queue at the instant t_1 (beginning of active period) when it just misses its turn to receive service. So, before receiving the first service it has to wait until an instant (say t'_1) while the other competing classes (C_2 and C_3) are served. This delay, $t'_1 - t_1 = X^h_{C_1}$, is maximized when competing classes consume their maximum credit [KS02]. The maximum credit that can be allocated to a class C_x in any DRR scheduling round is $Q^h_x + \Delta^{max,h}_x$. It belongs to the case where C_x consumed minimum amount of allocated credit in the previous round

so that it gets maximum deficit for the given round, which is also the case for C_2 and C_3 in Figure 2.5.

Since C_x is served as long as its remaining credit is not smaller than the size of the head-of-line frame in the C_x queue, the maximum deficit from any round is smaller than the largest C_x frame. Thus, we have

$$\Delta_{C_x}^{max,h} = l_{C_x}^{max,h} - 1 \tag{2.5}$$

So, in a DRR scheduler serving n traffic classes at an output port h, the maximum delay experienced by C_x flows before receiving the first service is:

$$X_{C_x}^h = \frac{\sum_{j=1,j\neq x}^{n} Q_{C_j}^h + \Delta_{C_j}^{max,h}}{R} \tag{2.6}$$

where R is the link rate.

This maximum delay is observed for class C_1 in round rd_1 as shown in Figure 2.5. In this round, both classes C_2 and C_3 consume their maximum credit (298 bytes) in the transmission of frames from v_6, v_7, v_8, v_{21}, v_{14} and v_{15}. This introduces a delay

$$\begin{aligned} t_1' - t_1 &= X_{C_1} \\ &= \frac{Q_{C_2} + \Delta_{C_2}^{max} + Q_{C_3} + \Delta_{C_3}^{max}}{R} \\ &= \frac{(199 + 100 + 199 + 100) \times 8}{100} \\ &= 47.68\ \mu sec \end{aligned}$$

for C_1 before it could receive the first service.

During the first service (from t_1' to t_2), in the worst case, C_1 consumes minimum credit, i.e. $Q_{C_1} - \Delta_{C_1}^{max} = 199 - 99 = 100$ bytes, which is the case when v_5 frame is served.

Since there is no deficit left for C_2 and C_3 in the first round, they get a credit of at most their quantum values (199 bytes) in the second round. C_2 and C_3 consume their full credit in the transmission of v_9, v_{13}, v_{16} and v_{17}. This leads to a scenario where C_1, since the beginning of service at t_1', is served at a service rate less than the theoretical service rate, i.e. from t_1' till the beginning of C_1 service in second round (at t_2''), we have:

$$\left(\frac{(Q_{C_1} - \Delta_{C_1}^{max})}{(Q_{C_1} - \Delta_{C_1}^{max}) + (Q_{C_2} + Q_{C_3})} \times R \right) < \rho_{C_1}$$

$$\left(\frac{100}{100+(199+199)} \times 100\right) < 33.33$$

so

$$20.08 < 33.33$$

This reduced service delays the interval, in which C_1 could be served at theoretical service rate, by Y_{C_1}. In the following round, C_1 will get the credit of at least $Q_{C_1} + \Delta_{C_1}^{max}$ bytes which permits the service of at least Q_{C_1} bytes. Hence, C_1 is served at its theoretical service rate starting from the service in the second round. This impact of reduced service is also shown in Figure 2.6 (left side), where Y_{C_1} is the interval between t_1' and an instant (say t_y). The instant t_y can be obtained graphically by extending the curve of slope 33.33 bits/μsec until it touches the x-axis, so that

$$Y_{C_1} = (t_2'' - t_1') - (t_2'' - t_y) = \frac{Q_{C_1} - \Delta_{C_1}^{max}}{20.08} - \frac{Q_{C_1} - \Delta_{C_1}^{max}}{33.33} = 39.84 - 24 = 15.84\ \mu sec$$

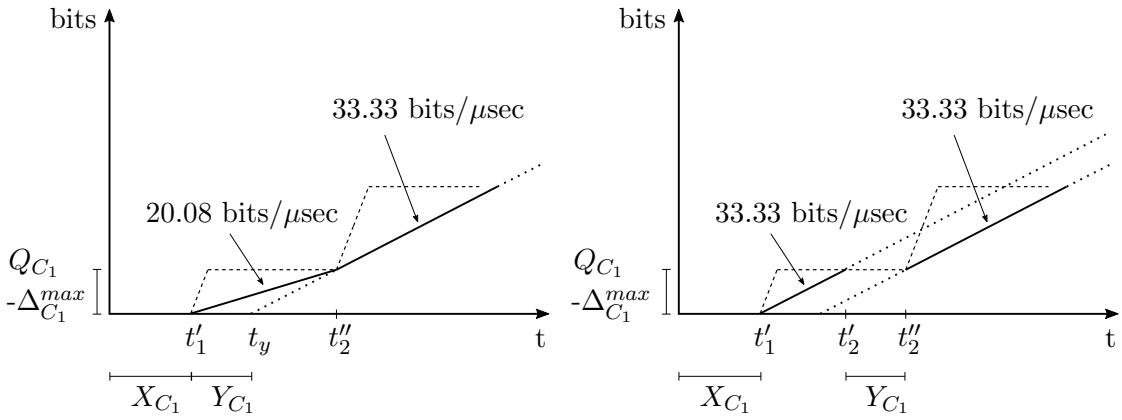


Figure 2.6 Illustration of the impact of reduced service in C_1

Another way to look at this interval of reduced service (from t_1' to t_2''), is the sum of two intervals (also shown in Figure 2.6 (right side)):

- interval t_1' to t_2', where C_1 receives service at theoretical service rate.
- interval t_2' to t_2'', where C_1 receives no service.

t_2' is somewhere between t_1' and t_2'', such that in interval t_1' to t_2' the ratio of the number of bytes served from C_1 and the sum of number of bytes served from all the active classes (including C_1) is same as the fraction of bandwidth utilized by C_1 if it were

served at its theoretical service rate. Hence, in interval t'_1 to t'_2, we have:

$$\frac{N_{C_1}}{N_{C_1} + N_{C_2} + N_{C_3}} = \frac{\rho_{C_1}}{R}$$

where $N_{C_1} (= Q_{C_1} - \Delta_{C_1}^{max})$, N_{C_2} and N_{C_3} are the number of bytes served from C_1, C_2 and C_3 respectively.

In the interval t'_2 to t''_2, no frames are served from C_1, which corresponds to the delay Y_{C_1}. These intervals are illustrated in Figure 2.4, 2.6 (right side) and 2.5.

This delay $Y_{C_x}^h$ for a class C_x in a node h is given by:

$$Y_{C_x}^h = \frac{(Q_{C_x}^h - \Delta_{C_x}^{max,h}) + \sum_{j=1, j\neq x}^{n} Q_{C_j}^h}{R} - \frac{(Q_{C_x}^h - \Delta_{C_x}^{max,h})}{\rho_{C_x}^h} \tag{2.7}$$

The first fraction computes the duration between t'_1 and t''_2, while the second one corresponds to the duration between t'_1 and t'_2. The delay $t''_2 - t'_2$ is the impact of the reduced service on class C_x.

Finally, the DRR scheduler latency is given by:

$$\Theta_{C_x}^h = X_{C_x}^h + Y_{C_x}^h \tag{2.8}$$

In Figure 2.5, the delay experienced by C_1 in round rd_2 due to reduced service is

$$\begin{aligned} Y_{C_1} &= \frac{(Q_{C_1} - \Delta_{C_1}^{max}) + Q_{C_2} + Q_{C_3}}{R} - \frac{(Q_{C_1} - \Delta_{C_1}^{max})}{\rho_{C_1}} \\ &= \frac{(100 + 199 + 199) \times 8}{100} - \frac{100 \times 8}{33.33} \\ &= 15.84\ \mu sec \end{aligned}$$

So the scheduler latency experienced by C_1 flows is $\Theta_{C_1} = 47.68 + 15.84 = 63.52\ \mu sec$. In the following rounds, C_1 is served at an average service rate equal to its theoretical service rate.

2.4.2 Modelling real-time switched Ethernet network with DRR scheduling in Network Calculus

In NC, a flow constrained by a maximum frame length l^{max} and a minimum inter-frame arrival time T at its source end-node is upper bounded by an arrival curve:

$$\alpha(t) = \frac{l^{max}}{T}t + l^{max}, \forall\ t > 0$$

In the given network, an end-node can be a source of one or more such flows. This end-node makes no assumption on the differentiation of flow traffic and emits each flow frame through the same FIFO multiplexer over the output link providing a service rate R bits/μsec . Thus, the service provided at this end-node is represented by a service curve $\beta(\mathrm{t}) = R[t]^{+}$.

The flows emitted by an end-node act as the incoming data stream in the network. The flows arriving at a switch output port h are differentiated based on their traffic classes and stored in their respective class buffers. The flow forwarding process from switch input towards output port buffer introduces a delay called switching latency which is upper bounded by a value sl. The arrival of flows in a class C_x queue is constrained by an overall arrival curve which is the cumulative curve of flows in this class queue:

$$\alpha^{h}_{C_x}(t) = \sum_{i \in \mathcal{F}^{h}_{C_x}} \alpha^{h}_{i}(t) \tag{2.9}$$

where $\mathcal{F}^{h}_{C_x}$ is the set of C_x flows and $\alpha^{h}_{i}(\mathrm{t})$ is the arrival curve of a flow v_i at h. This arrival curve includes the jitter J^{h}_{i} experienced by this flow in the network. This jitter is explained in Section 1.5.2.

As seen earlier, a DRR scheduler shares the maximum service capacity between the traffic classes based on the quantum assigned to each class. On the long term, the service rate received by C_x is lower bounded by the theoretical service rate $\rho^{h}_{C_x}$, which is a fraction of link rate R (equation (2.3)). Besides this lower service rate, a flow in C_x buffer could experience a scheduling latency ($\Theta^{h}_{C_x}$) given by equation (2.8). Thus, in the complete switching process from the entry till the exit of C_x frame at the switch, the service received by C_x frame is lower bounded in NC by a service curve:

$$\beta^{h}_{C_x} = \rho^{h}_{C_x}[t - sl - \Theta^{h}_{C_x}]^{+} \tag{2.10}$$

For the example in Figure 2.1, the service received by C_1 flows is shown in Figure 2.7

and is given by:

$$\beta_{C_1} = 33.33[t - 63.52 - sl]^+$$

Since a class alternates between being served and waiting for its chance to be served, the actual service received by this class is illustrated by a staircase curve in Figure 2.7. However, the NC employs only convex service curve, which is an under-approximation of the given staircase curve and is represented by equation (2.10).

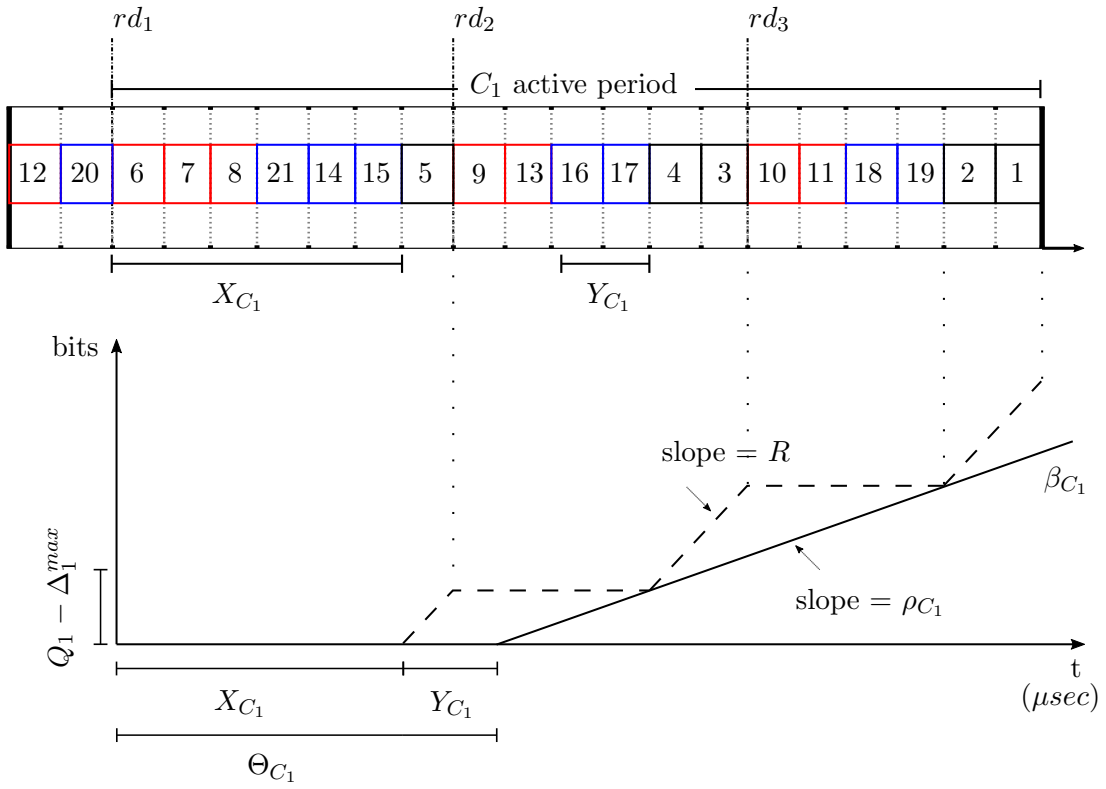


Figure 2.7 Worst-case service for C_1 in DRR scheduler

The worst-case delay of C_x flows is determined by the maximum horizontal distance $h(\alpha_{C_x}, \beta_{C_x})$ between its overall arrival curve and the service curve provided to these flows. This difference is given by:

$$D_i^h = \sup_{s \geq 0}(\inf\{\tau \geq 0 | \alpha_{C_x}^h(s) \leq \beta_{C_x}^h(s + \tau)\}) \tag{2.11}$$

The worst-case end-to-end delay in the network is the sum of worst-case delays experienced by the given flow v_i at each output port h that it traverses in the given path

$\mathcal{P}_i$. So, we have:

$$D_i^{E2E} = \sum_{h \in \mathcal{P}_i} D_i^h \tag{2.12}$$

A detailed illustration of end-to-end delay computation using the NC approach is presented with the help of a switched Ethernet network example in the next chapter (Section 3.3).

2.5 Conclusion

DRR scheduling is a QoS mechanism that can help to simplify network design by providing network sharing especially when different classes of flows need to be served. It provides guarantee on a share of bandwidth allocated to each class of flows, independently of the traffic in the other classes. But it adds an undeniable scheduler latency which is an important concern for the flows with strict timing constraints.

In this chapter, we have studied a QoS architecture in a switched Ethernet network based on DRR service discipline. The end-to-end delays experienced by flows in each traffic class are upper bounded by NC approach. In the next chapter, we will see that these delays are affected by the quantum assigned to each class. Thus, the choice of this quantum is crucial to guarantee the delay constraints in critical flow classes. We will propose an algorithm for the choice of quantum and show that it can improve overall performance of the network.

Chapter 3

Efficient Resource Sharing in a QoS-Aware Switched Ethernet Network

Contents

3.1 Introduction 59
3.2 Assumptions 60
3.3 Delay computation by classical NC approach 61
3.4 Quantum assignment 67
3.4.1 Properties of classical NC approach 68
3.4.2 Optimal Quantum assignment algorithm 74
3.5 Conclusion 78

3.1 Introduction

In the previous chapter, we have seen that the DRR scheduling allows guarantee on a long-term bandwidth available to flow classes sharing the network. In a network serving mixed-criticality flows, such a mechanism can be utilised, first, to better use the bandwidth for critical flows assigned to classes with very different timing constraints, second, to allow the transmission of less/non critical flows.

The guarantee on worst-case delays in this network can be provided by Boyer's NC approach by upper bounding the end-to-end delays. This approach was also illustrated in the previous chapter (Section 2.4). We call this approach the "classical NC approach".

The main goal of this chapter is thus to propose a quantum assignment algorithm that guarantees a delay upper limit to the critical flows having strict delay constraints and also allows to improve the service provided to non-critical flows. We will see that the classical NC approach computes the delay bounds for each class independently and also that the computation for a given class only depends on the quantum assigned to the class and the sum of all the quanta. This property will limit the complexity of the quantum assignment algorithm.

In Section 3.2 we present network architecture and the assumptions specific to this chapter. In Section 3.3 we illustrate the delay computation using classical NC approach. In Section 3.4 we propose our quantum assignment algorithm. Section 3.5 concludes the chapter.

3.2 Assumptions

The network model considered in this chapter is the same as the one given in the previous chapter but with some additional assumptions. We consider a network shared by flows of different level of criticality (safety-critical, less-critical, best-effort) with different delay constraints. As discussed earlier in Section 1.8, one way to distribute these flows in classes can be based on the range of their deadlines. So, we assume (n-1) classes of critical flows with different delay constraint per class and 1 class for non-critical flows which do not have the strict delay constraint. The distribution of flows in $n-1$ critical classes is in increasing order of deadlines where C_1 flows have the smallest deadline. Figure 3.1 illustrates this kind of network architecture.

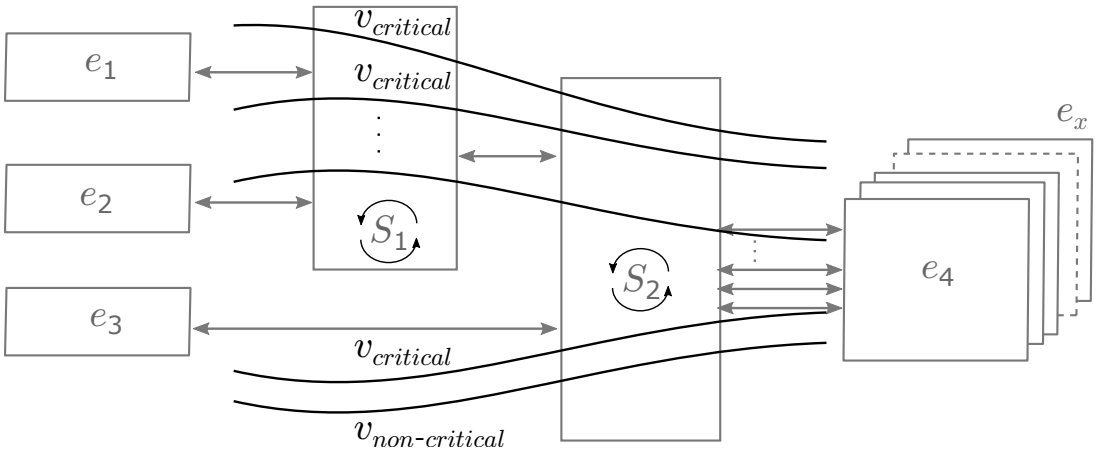


Figure 3.1 Network architecture

Each switch implements DRR scheduler at its output port.

A sum of quanta Q is assigned to each switch output port. Since there are n traffic classes, these quanta are distributed among each class such that:

$$Q = \sum_{1 \leq j \leq n} Q_{C_j} \tag{3.1}$$

We assume that the sum of quanta as well as the distribution between classes is the same for all the switch output ports. Therefore, the symbol h (that represents a switch output port) can be omitted.

3.3 Delay computation by classical NC approach

As shown in Section 2.4, the worst-case delay of class C_x flows in a given network element, computed by classical NC approach, is determined by the maximum horizontal distance $h(\alpha_{C_x}, \beta_{C_x})$ between its overall arrival curve and the service curve. The arrival curve α_{C_x} represents the cumulative traffic of flows of C_x arriving at the given network element and the service curve β_{C_x} represents the lower bound on the service provided to C_x. Thus for a given amount of traffic from C_x flows the delay computation is affected by the service available to these flows.

In the following paragraphs, we illustrate how the choice of quantum affects the service available to a class and thus its delay.

Let us consider the network example shown in Figure 3.2 with the flow and class configuration given by Table 3.1. The flows are divided into two critical classes (C_1 and C_2) and one non-critical class (C_3), where critical classes have a strict delay constraint. The maximum service rate at each output port is $R = 100$ bits/μsec and switching latency is assumed to be null. In this example, we focus only on end-to-end delay computation of flow v_2 of class C_1 in its path e_2–S_1–S_2–e_7.

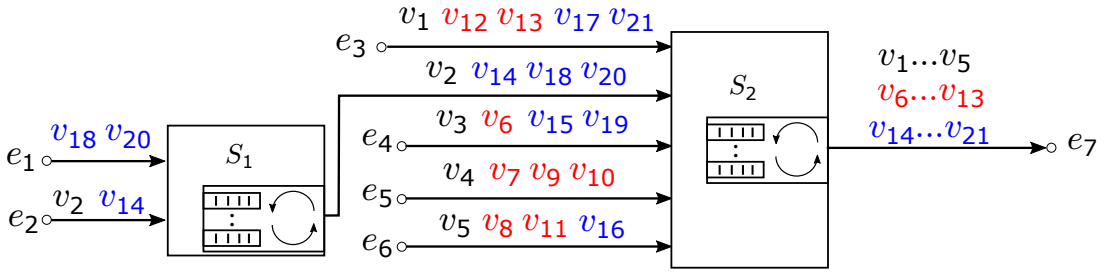


Figure 3.2 A switched Ethernet network example

Table 3.1 Flow and class specifications

Flows	T_i (msec)	Frame lengths (bytes)
$v_6, v_{12}, v_{13}, v_{20}$	128	80–100
$v_1, v_7, v_8, v_9, v_{17}$	128	80–99
$v_2, v_4, v_5, v_{10}, v_{16}, v_{18}, v_{21}$	64	80–100
$v_3, v_{11}, v_{14}, v_{15}, v_{19}$	64	80–99

Flows	Deadline (μsec)	Class
v_1 — v_5	250	Critical (C1)
v_6 — v_{13}	500	Critical (C2)
v_{14} — v_{21}	-	Non-Critical (C3)

Let us analyse the following three cases:

case 1: The sum of quanta Q of 655 bytes is distributed among each class such that $Q_{C_1} = 268$, $Q_{C_2} = 103$ and $Q_{C_3} = 284$ bytes.

At end-node e_2, since there is no flow differentiation mechanism the overall arrival curve $\alpha_o^{e_2}$ is the sum of arrival curves of all the flows sharing the output port and the service curve for any flow emitted by this end-node is $\beta^{e_2} = R[t]^+ = 100[t]^+$. These curves are shown in Figure 3.3 (left). Thus, the worst-case delay for v_2 is

$$D_{v_2}^{e_2} = h(\alpha_o^{e_2}, \beta^{e_2})$$

where,

$$\begin{aligned}
\alpha_o^{e_2} &= \alpha_{v_2}^{e_2} + \alpha_{v_{14}}^{e_2} \\
&= \left(\frac{l_{v_2}^{max}}{T_{v_2}}\, t + l_{v_2}^{max}\right) + \left(\frac{l_{v_{14}}^{max}}{T_{v_{14}}}\, t + l_{v_{14}}^{max}\right) \\
&= \left(\frac{100 \times 8}{64000}\, t + (100 \times 8)\right) + \left(\frac{99 \times 8}{64000}\, t + (99 \times 8)\right) \\
&= 0.024\, t + 1592
\end{aligned}$$

so, we get $D_{v_2}^{e_2} = 15.92$ μsec.

At switch S_1 and S_2, the flow v_2 is classified as C_1 by the DRR scheduler. Since the service provided at a switch output port is shared by flows of each class in each DRR scheduling round, these flows alternate between being served and waiting for their turn to be served. So, the long-term service rate (from equation (2.3)) received by C_1 flows at

each switch output port is at least

$$\begin{aligned}\rho_{C_1} &= \frac{Q_{C_1}}{Q_{C_1} + Q_{C_2} + Q_{C_3}} \times R \\ &= \frac{268}{268 + 103 + 284} \times 100 \\ &= 40.91\ bits/\mu sec\end{aligned}$$

and the scheduler latency (from equation (2.8)) is

$$\Theta_{C_1} = X_{C_1} + Y_{C_1}$$

where,

$$\begin{aligned}X_{C_1} &= \frac{Q_{C_2} + \Delta_{C_2}^{max} + Q_{C_3} + \Delta_{C_3}^{max}}{R} \\ &= \frac{(103 + 99 + 284 + 99) \times 8}{100} \\ &= 46.8\ \mu sec\end{aligned}$$

and

$$\begin{aligned}Y_{C_1} &= \frac{(Q_{C_1} - \Delta_{C_1}^{max}) + Q_{C_2} + Q_{C_3}}{R} - \frac{(Q_{C_1} - \Delta_{C_1}^{max})}{\rho_{C_1}} \\ &= \frac{(169 + 103 + 284) \times 8}{100} - \frac{169 \times 8}{40.91} \\ &= 11.44\ \mu sec\end{aligned}$$

so,

$$\Theta_{C_1} = 46.8 + 11.44 = 58.24\ \mu sec$$

So, based on the classical NC approach, the residual service (from equation (2.10)) for C_1 flows is given as

$$\beta_{C_1} = \rho_{C_1}[t - \Theta_{C_1}]^+ = 40.91 \times [t - 58.24]^+$$

The delay experienced by C_1 flows constrained by an overall arrival curve $\alpha_{C_1}(\mathrm{t})$ in a switch output port offering a service curve $\beta_{C_1}(\mathrm{t})$ is bounded by the maximum horizontal difference (from equation (2.11)) between these curves. Since we have assumed same

scheduler configuration at each switch output port, the service curve β_{C_1}(t) is also same on these ports whereas the overall arrival curve (from equation (2.9)) depends upon the flows sharing the C_1 buffer at each output port. For instance, there is only one flow v_2 from class C_1 at S_1 so the overall arrival curve is equal to the arrival curve of v_2

$$\begin{aligned}\alpha_{C_1}^{S_1} &= \alpha_{v_2}^{S_1} \\ &= \frac{l_{v_2}^{max}}{T_{v_2}}(t + J_{v_2}^{S_1}) + l_{v_2}^{max} \\ &= 0.0125\ (t + 7.92) + 800\end{aligned}$$

and the overall arrival curve at S_2 is the cumulative curve of v_1, v_2, v_3, v_4, and v_5 which is given as

$$\begin{aligned}\alpha_{C_1}^{S_2} &= \sum_{i=1}^{5} \alpha_{v_i}^{S_2} \\ &= \sum_{i=1}^{5} \left(\frac{l_{v_i}^{max}}{T_{v_i}}(t + J_{v_i}^{S_2}) + l_{v_i}^{max} \right) \\ &= 0.056\ t + 3986.07\end{aligned}$$

Thus, the worst-case delays for v_2 in these output ports are $D_{v_2}^{S_1} = h(\alpha_{C_1}^{S_1}, \beta_{C_1}) = 77.79$ μsec (Figure 3.3 (center)) and $D_{v_2}^{S_2} = h(\alpha_{C_1}^{S_2}, \beta_{C_1}) = 155.65$ μsec (Figure 3.3 (right)).

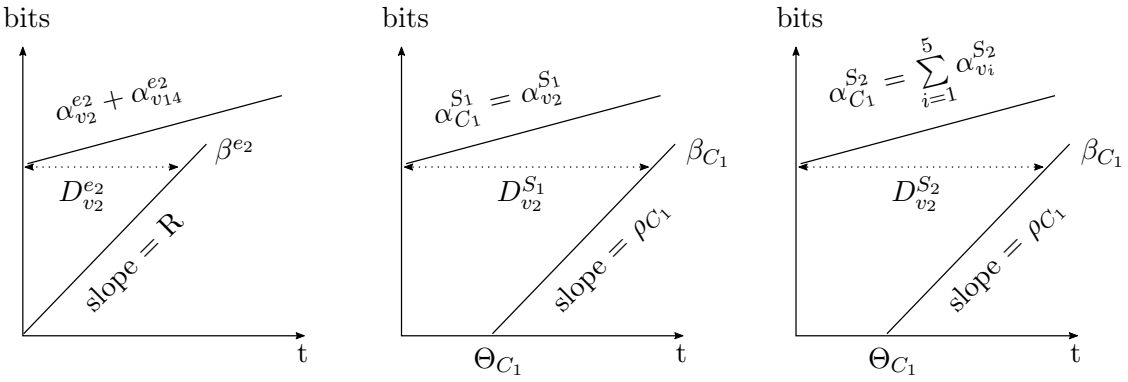


Figure 3.3 Delay computation for flow v_2 of C_1

Based on the NC approach, the worst-case end-to-end delay for a flow is the sum of worst-case delays at each output port in the path followed by this flows. So, the end-to-end delay for v_2 is $D_{v_2}^{E2E} = 15.92 + 77.79 + 155.65 = 249.36$ μsec.

case 2: The same sum of quanta $Q = 655$ bytes is distributed among each class as $Q_{C_1} = 169$, $Q_{C_2} = 202$ and $Q_{C_3} = 284$.

At e_2, since the delay computation does not depend upon the quantum, we have $D_{v_2}^{e_2}$ $= 15.92$ μsec.

At the switches S_1 and S_2, since the arrival curve computation does not depend upon quantum value it is not affected, the only affect can be due to the change in jitter. In the computation of the service curve, the given distribution of quanta leads to a long-term service rate and a scheduler latency for C_1 flows as:

$$\begin{aligned}
\rho_{C_1} &= \frac{Q_{C_1}}{Q_{C_1} + Q_{C_2} + Q_{C_3}} \times R \\
&= \frac{169}{169 + 202 + 284} \times 100 \\
&= 25.8\ bits/\mu sec \\
\Theta_{C_1} &= X_{C_1} + Y_{C_1} \\
&= \left(\frac{Q_{C_2} + \Delta_{C_2}^{max} + Q_{C_3} + \Delta_{C_3}^{max}}{R} \right) \\
&+ \left(\frac{(Q_{C_1} - \Delta_{C_1}^{max}) + Q_{C_2} + Q_{C_3}}{R} - \frac{(Q_{C_1} - \Delta_{C_1}^{max})}{\rho_{C_1}} \right) \\
&= \left(\frac{(202 + 99 + 284 + 99) \times 8}{100} \right) \\
&+ \left(\frac{(70 + 202 + 284) \times 8}{100} - \frac{70 \times 8}{25.8} \right) \\
&= 54.72 + 22.78 = 77.5\ \mu sec
\end{aligned}$$

so, the service curve for C_1 flow is

$$\beta_{C_1} = 25.8\ [t - 77.5]^+$$

In this case, since the service rate is reduced and the scheduler latency is increased for C_1 flows as compared to the previous one, the delay at each switch output port is, not surprisingly, increased to $D_{v_2}^{S_1} = 108.51$ μsec and $D_{v_2}^{S_2} = 232$ μsec. Therefore, the end-to-end delay for v_2 is $D_{v_2}^{E2E} = 356.43$ μsec.

case 3: The sum of quanta is increased to $Q = 1310$ bytes but its distribution among each class is proportional to the one in first case, i.e. $Q_{C_1} = 268{\times}2 = 536$, $Q_{C_2} = 103{\times}2$ $= 206$ and $Q_{C_3} = 284{\times}2 = 568$ bytes.

So, the service curve for C_1 flows at each switch output port is

$$\begin{aligned}
\rho_{C_1} &= \frac{Q_{C_1}}{Q_{C_1} + Q_{C_2} + Q_{C_3}} \times R \\
&= \frac{536}{536 + 206 + 568} \times 100 \\
&= 40.91 \; bits/\mu sec \\
\Theta_{C_1} &= X_{C_1} + Y_{C_1} \\
&= \left(\frac{(206 + 99 + 568 + 99) \times 8}{100} \right) \\
&+ \left(\frac{(437 + 206 + 568) \times 8}{100} - \frac{437 \times 8}{40.91} \right) \\
&= 77.76 + 11.43 = 89.19 \; \mu sec \\
\beta_{C_1} &= 40.91 \; [t - 89.19]^+
\end{aligned}$$

In this case, the service rate for C_1 flows is the same as the one in the first case but the scheduler latency is increased. So the delay at each switch output port is also increased to $D_{v_2}^{S_1} = 108.75$ μsec and $D_{v_2}^{S_2} = 186.63$ μsec. Therefore, the end-to-end delay for v_2 is $D_{v_2}^{E2E} = 311.3$ μsec.

These three cases are summarised in Table 3.2. They illustrate that the sum of quanta distributed among the classes and the quantum assigned to an individual class has a direct effect on the service received by this class and eventually on the delay experienced by this class flows. In the following paragraphs, we derive some properties to show the relationship between the quantum assignment and the worst-case delay bounds and then we propose quantum assignment algorithm.

Table 3.2 Delay variation with quanta

Case	(bytes)				bits/μsec	(μsec)	
	Q	Q_{C_1}	Q_{C_2}	Q_{C_3}	ρ_{C_1}	Θ_{C_1}	$D_{v_2}^{E2E}$
1	655	268	103	284	40.91	58.24	249.36
2	655	169	202	284	25.8	77.5	356.43
3	1310	536	206	568	40.91	89.19	311.3

3.4 Quantum assignment

In a network serving flows having different delay constraints, the service provided to these flows must allow to deliver them to their destination end-node within their deadlines. Such service is assured when the delay upper bounds of these flows are less than their deadlines.

We have seen through an example in the previous section that the delay bound of a flow is affected by the quantum value. In the first case of the example, the distribution of $Q = 655$ among the classes led to an end-to-end delay bound $D_{v_2}^{E2E} = 249.36$ μsec, for a critical flow v_2, which is less than its deadline of 250 μsec. Whereas in the second case, where we distributed the same sum of quanta in another proportion, the end-to-end delay bound for v_2 is increased to a value (356.43 μsec) which is higher than its deadline. Furthermore, in the third case, where we distributed another value of $Q = 1310$ such that the portion (in %) of Q available to each class is same as the one in the first case, the end-to-end delay bound (311.3 μsec) is again higher than the v_2 deadline. The sum Q of quanta can be considered either valid or invalid based on the possibility to distribute it among the classes such that it assures the delay constraints for the flows of these classes. Besides, the distribution of quanta is also constrained by the fact that each class should receive a quantum that allows transmission of a frame of any size in the class buffer. Therefore, we define a valid value of Q as follows.

Definition 4. *A sum of quanta (Q) is valid if there exists at least a distribution such that each class C_x gets a quantum greater than or equal to its largest frame size ($l_{C_x}^{max}$), i.e. the quantum assigned to C_x satisfies equation (2.2), and the maximum end-to-end delay for each flow is within its delay constraint.*

Based on definition 4, there can be multiple values of Q which are valid for a given network configuration. For each valid Q there exist one or more distributions that result in the end-to-end delay bounds which are less than the flow deadlines.

Since we have considered a network architecture with (n-1) classes of critical flows having different deadlines and 1 class for non-critical flows having no delay constraints, we define an optimal value of Q and its distribution as a valid one which corresponds to the scenario where the delay in non-critical flows is minimum. Indeed, for a given value of Q, when the quantum share for a class is reduced it leads to a lower bandwidth for this class (see equation (2.3)) and leaves a higher residual bandwidth for other classes which can result in reducing their delays. Therefore, we define an optimal Q as follows.

Definition 5. *A value of Q is optimal if it leads to a distribution that:*

- *guarantees that the critical flows respect their delay constraints, i.e. satisfy Definition 4, and*
- *maximise the bandwidth available to non-critical flows.*

It should be noted that the optimality of the solution depends on the WCTT analysis. In our case, this analysis is based on the classical NC approach, which is pessimistic. Due to this pessimism, a valid distribution might be far from an optimal one. In that case the value of Q is not optimal. In the rest of this thesis, the optimal Q means with respect to the WCTT analysis based on the classical NC approach.

We will see in the next section that this optimal Q corresponds to the lowest valid Q. In order to find the optimal Q, we propose an algorithm in Section 3.4.2. This algorithm proceeds in multiple iterations. In each iteration, for a given value of Q, we find a minimum quantum (that leads to an end-to-end delay bound which is less than but close to the specified deadline) for each critical class so that a maximum residual quantum is available to the non-critical class. The algorithm works only when it is possible to compute the delay for each class independently. We will show in the following paragraphs that by using the classical NC approach it is possible to compute the maximum delay for the flows in each class independently. For this purpose, we establish some properties of the classical NC approach. These properties state that, first, the delay computation for a given class is not affected by a change in quantum distribution among concurrent classes and, second, that the optimal quanta correspond to the lowest value of valid quanta.

3.4.1 Properties of classical NC approach

In this section, we present the properties that serve as the basis for our quantum assignment algorithm in Section 3.4.2.

First, we show that, for a class C_x, the worst-case end-to-end delay computed by the classical NC approach depends upon the quantum Q_{C_x} assigned to C_x and the sum of quanta Q but not on the distribution of $Q - Q_{C_x}$ among other classes.

Property 1. *Given a Q_{C_x} and a Q, the worst-case delay for C_x computed by the classical NC approach is the same for any quantum distribution among competing classes C_j $(j \neq x)$.*

Proof. The quanta impact the service curve for class C_x in each switch output port, but not the arrival curve. So, we will prove that this service curve depends on Q_{C_x} and Q, but not on quantum distribution.

The service curve for class C_x is defined as:

$$\beta_{C_x} = \rho_{C_x}[t - sl - \Theta_{C_x}]^+$$

where sl is constant. So, β_{C_x} depends on the bandwidth ρ_{C_x} and the scheduler latency Θ_{C_x}.

The bandwidth ρ_{C_x} allocated to C_x is the fraction of link bandwidth R, which is defined as

$$\rho_{C_x} = \frac{Q_{C_x}}{Q} \times R$$

Thus ρ_{C_x} only depends on Q_{C_x} and Q.

The scheduler latency Θ_{C_x} experienced by C_x flows is the sum of two delays X_{C_x} and Y_{C_x}, which are defined as,

$$X_{C_x} = \frac{1}{R}\left(\sum_{\substack{1\le j\le n\\ j\neq x}} (Q_{C_j} + \Delta_{C_j}^{max})\right)$$

$$Y_{C_x} = \frac{1}{R}\left(Q_{C_x} - \Delta_{C_x}^{max} + \sum_{\substack{1\le j\le n\\ j\neq x}} Q_{C_j} - \frac{Q_{C_x} - \Delta_{C_x}^{max}}{\frac{Q_{C_x}}{Q}}\right)$$

Thus, Θ_{C_x} can be written as

$$\Theta_{C_x} = X_{C_x} + Y_{C_x}$$

$$or \quad \frac{1}{R}\left(2 \times \sum_{\substack{1\le j\le n\\ j\neq x}} Q_{C_j} + \sum_{\substack{1\le j\le n\\ j\neq x}} \Delta_{C_j}^{max} + (Q_{C_x} - \Delta_{C_x}^{max})\left(1 - \frac{Q}{Q_{C_x}}\right)\right)$$

$$or \quad \frac{1}{R}\left(2 \times \sum_{\substack{1\le j\le n\\ j\neq x}} Q_{C_j} + \sum_{\substack{1\le j\le n\\ j\neq x}} \Delta_{C_j}^{max} + (Q_{C_x} - \Delta_{C_x}^{max})\left(\frac{Q_{C_x} - Q}{Q_{C_x}}\right)\right)$$

$$or \quad \frac{1}{R}\left(2 \times (Q - Q_{C_x}) + \sum_{\substack{1\le j\le n\\ j\neq x}} \Delta_{C_j}^{max} - (Q_{C_x} - \Delta_{C_x}^{max})\left(\frac{Q - Q_{C_x}}{Q_{C_x}}\right)\right)$$

$$\left[since \sum_{\substack{1 \leq j \leq n \\ j \neq x}} Q_{C_j} = Q - Q_{C_x} \right]$$

$$or \quad \frac{1}{R} \left((Q - Q_{C_x}) + \sum_{\substack{1 \leq j \leq n \\ j \neq x}} \Delta_{C_j}^{max} + \Delta_{C_x}^{max} \left(\frac{Q - Q_{C_x}}{Q_{C_x}} \right) \right)$$

Therefore, we can conclude that, Θ_{C_x} depends on

- the link rate R and the maximum deficit $\Delta_{C_i}^{max}$ $(= l_{C_i}^{max} - 1)$ for each class, which is constant,
- the quantum Q_{C_x} assigned to class C_x under study,
- the value $(Q - Q_{C_x})$ assigned to competing classes C_j $(j \neq x)$, but not the individual Q_{C_j} values.

□

Based on Property 1 it is possible to compute the delay for a given class C_x independently and this delay remains the same for any value of quantum assigned to concurrent classes C_j $(j \neq x)$ as long as the sum of quanta Q remains unchanged. Such property allows to compute, independently, the quantum required by each critical class to satisfy its delay constraint. Therefore, the quantum assignment to $n-1$ critical classes can be done in the following manner.

- the value of Q to be distributed among classes is fixed,
- we calculate the minimum portion of Q which should be assigned to C_1, i.e. Q_{C_1}, such that all the flows in C_1 respect their deadlines,
- we do the same for $C_2 \ldots C_{n-1}$.

Now, we show that the minimum quantum Q_{C_x} required to satisfy the delay constraint of C_x is not the same for different values of Q. For that purpose, let us come back to our network example shown in Figure 3.2 and consider the following scenarios:

case 1: If $Q = 600$ bytes, on computing the delay bound with NC approach we observe that the minimum quantum required by C_1 to respect its delay constraint is 240 bytes (40% of Q) and that for C_2 is 92 bytes (15.33 % of Q). So, the residual quantum

for C_3 is 268 bytes (which is 44.67% of Q). However, this distribution does not satisfy all the condition of Definition 4 since the quantum assigned to C_2 is less than $l_{C_2}^{max} =$ 100 bytes and, thus, is not a valid.

case 2: Now, let us assume a higher value of $Q = 670$ bytes. In this case, the minimum quantum required by C_1 is 276 bytes (41.19% of Q) and that for C_2 is 106 bytes (15.82% of Q). So, the residual quantum for C_3 is 288 bytes (42.99% of Q). The percentage share of quanta required by critical classes C_1 and C_2 is clearly changed as compared to the previous case but it was not a valid case. In the next case, let us try a value of Q somewhere between that in case 1 and case 2.

case 3: Let $Q = 655$ bytes. In this case, the minimum quantum required by C_1 is 268 bytes (40.92% of Q) and that for C_2 is 103 bytes (15.72% of Q), which gives the residual quantum for C_3 as 284 bytes (43.35% of Q).

On comparing the two valid cases (2 and 3), it is clear that the minimum portion of Q required by critical classes to respect their deadlines is not the same for different value of Q.

Next, we identify the value of Q that leads to maximum bandwidth share for flows in non-critical class (C_n). Since the quantum share required by critical classes is not the same for all values of Q, we want to get the valid Q that maximises the percentage of Q which is not assigned to critical classes.

In the next property, we establish that the best Q value is the smallest valid one.

Property 2. *When each critical class is assigned the minimum quantum which guarantees that no deadline is missed (based on the delay bounds obtained from the classical NC approach), a smaller valid value of Q never leads to a smaller percentage of Q assigned to the non-critical class C_n.*

Proof. Since, we have $Q_{C_n} = Q - \sum_{1 \leq j < n} Q_{C_j}$, the percentage of Q assigned to C_n can be given by

$$\left(1 - \sum_{1 \leq j < n} \frac{Q_j}{Q}\right) \times 100\%$$

Let's consider two valid Q values: Q' and Q'', where $Q'' = \sigma \times Q'$ with $\sigma > 1$. For the Property 2 to be true, the percentage of Q' assigned to C_n cannot be smaller than the percentage of Q'' assigned to C_n, i.e.

$$\sum_{1 \leq j < n} \frac{Q'_{C_j}}{Q'} \leq \sum_{1 \leq j < n} \frac{Q''_{C_j}}{Q''}$$

This inequality is true when

$$\frac{Q'_{C_j}}{Q'} \leq \frac{Q''_{C_j}}{\sigma \times Q'} \quad for \quad 1 \leq j < n \tag{3.2}$$

Let's consider one class C_x $(1 \leq x < n)$. When the sum of quanta is Q', the class C_x should get a quantum Q'_{C_x}, which is the minimum quantum it needs to meet all its deadlines. And when the sum of quanta is $Q'' = \sigma \times Q'$, in this case the minimum quantum required by C_x must be

$$Q''_{C_x} \geq \sigma \times Q'_{C_x} \tag{3.3}$$

By the NC approach, the worst-case delay of a C_x flows is the maximum horizontal distance between overall arrival curve (α_{C_x}) of C_x flows and the residual service curve (β_{C_x}) offered to these flows. We already know that the arrival curve does not depend on quantum. For a given value of quanta Q'', the distance between α_{C_x} and β_{C_x} clearly decreases when Q''_{C_x} increases (since, $\rho_{C_x} = \frac{Q_{C_x}}{\sum\limits_{1 \leq j \leq n} Q_{C_j}} \times R$). Thus, the equation (3.3) is true if moving from a quantum of Q'_{C_x} out of Q' to a quantum of $\sigma \times Q'_{C_x}$ out of $\sigma \times Q'$ does not lead to a higher service curve for C_x.

The service curve (β_{C_x}) of C_x depends upon the fraction ρ_{C_x} of bandwidth allocated to C_x and the scheduler latency Θ_{C_x}.

For the given two valid quantum values Q' and Q'', the fraction of bandwidth allocated to C_x are

$$\rho'_{C_x} = \frac{Q'_{C_x}}{Q'} \times R \quad and \quad \rho''_{C_x} = \frac{Q''_{C_x}}{Q''} = \frac{\sigma \times Q'_{C_x}}{\sigma \times Q'} \times R$$

hence $\rho'_x = \rho''_x$.

In the proof of Property 1 we have seen that

$$\Theta'_{C_x} = \frac{1}{R}\left((Q' - Q'_{C_x}) + \sum_{\substack{1 \leq j \leq n \\ j \neq x}} \Delta^{max}_{C_j} + \Delta^{max}_{C_x}\left(\frac{Q' - Q'_{C_x}}{Q'_{C_x}}\right)\right)$$

For $Q'' = \sigma \times Q'$ and $Q''_{C_x} = \sigma \times Q'_{C_x}$, we have

$$\Theta''_{C_x} = \frac{1}{R}\left((\sigma \times Q' - \sigma \times Q'_{C_x}) + \sum_{\substack{1 \le j \le n \\ j \neq x}} \Delta^{max}_{C_j} + \Delta^{max}_{C_x}\left(\frac{\sigma \times Q' - \sigma \times Q'_{C_x}}{\sigma \times Q'_{C_x}}\right)\right)$$

$$= \frac{1}{R}\left(\sigma(Q' - Q'_{C_x}) + \sum_{\substack{1 \le j \le n \\ j \neq x}} \Delta^{max}_{C_j} + \Delta^{max}_{C_x}\left(\frac{Q' - Q'_{C_x}}{Q'_{C_x}}\right)\right)$$

hence, for $\sigma > 1$, we have $\Theta''_{C_x} > \Theta'_{C_x}$.

Thus the service curve for C_x when considering Q'' and Q''_{C_x} is under the service curve for C_x when considering Q'_{C_x} and Q'. Therefore, the eqiation (3.3) is true.

This phenomenon is illustrated in Figure 3.4. When σ increases, the service curve is shifted to the right, leading to a higher horizontal distance with traffic curve and, consequently, a higher worst-case delay. □

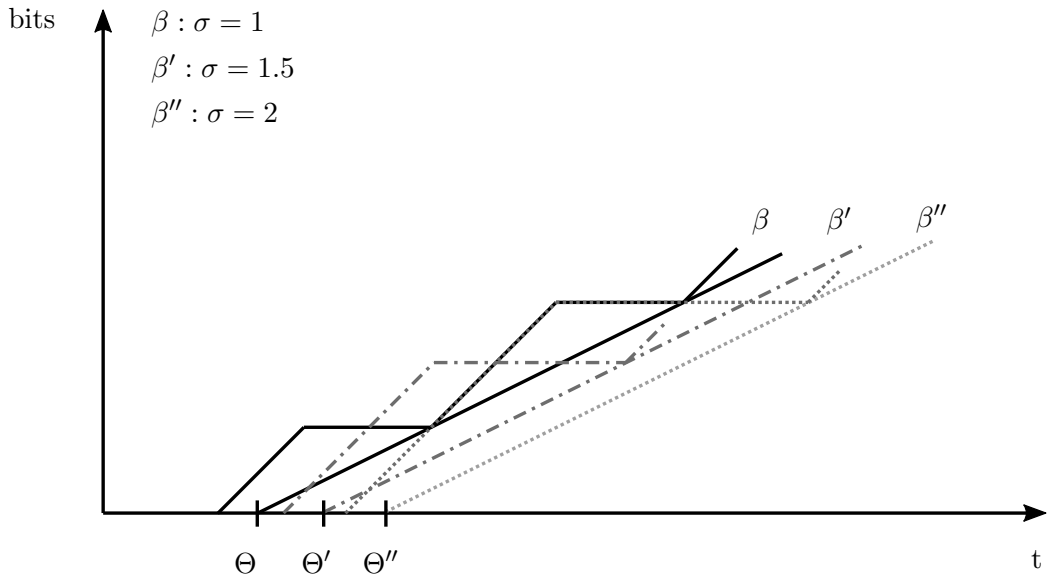


Figure 3.4 Service curves

Based on the Property 1 and 2, we propose an algorithm to find the optimal value of Q in the following paragraphs.

3.4.2 Optimal Quantum assignment algorithm

The basic idea of Algorithm 2 is to find an optimal Q by a validation and improvement technique. It starts with an initial value of Q. This Q is declared valid if it satisfies Definition 4, i.e. a successful distribution of Q among critical classes is found so that the end-to-end delay bound of each critical flow is less than its deadline and the portion of Q available to each class is greater than or equal to its maximum frame size. Such distribution of Q is found with the help of classical NC approach, thanks to Property 1. Then, the valid Q is considered either improvable or optimal. A valid Q is optimal when it leads to maximum residual bandwidth for non-critical class (Definition 5) and since the residual bandwidth for a non-critical class is higher at a lower value of Q (Property 2), an optimal Q is the smallest valid Q. When the value of Q is improvable, the process is repeated with a new (improved) value of Q. Whereas when the Q is not valid, an attempt is made to find a valid Q.

Let us have a look at the steps followed in Algorithm 2.

The process starts with an initial value $Q = \mathit{ValInit}$ (line 1) for which we compute the minimum portion Q_{C_i} of Q required by flows in each critical class C_i to respect their deadlines (line 9). The process stops as soon as the unused portion ($QResid$) of Q is less than the required value Q_{C_i} (line 10-11) or all the critical classes have been treated (line 8). In the former case, no valid solution can be found (line 11, 19-20, 34-35). In the latter case, the non-critical class gets the residual quantum $QResid$ (line 22) and a solution has been obtained. This solution might be either not valid (line 25) if equation 2.2 is not respected (at least one class has a quantum which is smaller than its maximum frame size) or improvable (line 25) if the quantum of every class exceeds its maximum frame size by a factor greater than a configured (small) value ϵ: we tolerate an ϵ difference since it might be tricky to reach a valid solution where at least one class has its maximum frame size as quantum. In both cases (not valid or improvable solution), the algorithm adapts the sum of quanta Q (lines 25-26) as: increase Q if the solution is not valid, reduce Q if the solution is improvable. As soon as a non-improvable valid solution is found (line 27-28), the algorithm stops.

The function *QuantumMin(i,Q)* (line 9) implements a binary search. It utilises the classical NC approach to compute the smallest Q_{C_i} which corresponds to no-deadline-miss condition for C_i.

Let us illustrate the operation of Algorithm 2 by computing the optimal Q and its distribution for the network example given in Figure 3.2.

The flows are divided into $n = 3$ classes based on their delay constrains. C_1 and C_2 are critical flow classes with delay constraint 250 μsec and 500 μsec, respectively, and

Algorithm 2: Quantum assignment algorithm

input : Initial sum of quanta: $ValInit$ (Integer)
input : Per class maximum frame size: $l_{C_1}^{max} \dots l_{C_n}^{max}$ (Integer)
input : Improvement level: ϵ (Float)
output : Valid solution found: $Valid$ (Boolean)
output : Per class quantum: $Q_{C_1} \dots Q_{C_n}$ (Integer)
Data: Remaining quantum: $QResid$
Data: Relative distance to a non improvable solution: min (Float)
Data: End of the process: $Fini$ (boolean)
Data: Counter: i (Integer)

$Q \leftarrow ValInit$;
$Fini \leftarrow false$;
while *not* $Fini$ **do**
  $i \leftarrow 1$;
  $QResid \leftarrow Q$;
  $Valid \leftarrow true$;
  $min \leftarrow Q$;
  while $i < n$ ***and*** $Valid$ **do**
    $Q_{C_i} \leftarrow QuantumMin(i, Q)$;
    if $Q_{C_i} > QResid$ **then**
      $Valid \leftarrow false$;
    else
      $QResid \leftarrow QResid - Q_i$;
      if $min > \frac{Q_{C_i}}{l_{C_i}^{max}}$ **then**
        $min \leftarrow \frac{Q_{C_i}}{l_{C_i}^{max}}$;
      $i \leftarrow i + 1$;
    end
  end
  if *not* $Valid$ **then**
    $Fini \leftarrow true$;
  else
    $Q_{C_n} \leftarrow QResid$;
    if $min > \frac{Q_n}{l_{C_n}^{max}}$ **then**
      $min \leftarrow \frac{Q_n}{l_{C_n}^{max}}$;
    if $min < 1$ ***or*** $min > 1 + \epsilon$ **then**
      $Q \leftarrow \frac{Q}{min}$;
    else
      $Fini \leftarrow true$;
    end
  end
end
if *valid* **then**
  The non improvable solution is reached ;
else
  No valid solution was found ;
end

C_3 is non-critical flow class with no constraint on delay (see Table 3.3). The maximum service rate at each output port is $R = 100$ bits/μsec.

Initially, we assume $Q = ValInit = 646$ bytes. So, $QResid = 646$ bytes (line 5), $Valid = true$ (line 6) and $min = 646$ (line 7). First critical class is C_1. Based on the given value of Q, the function *QuantumMin(C_1,Q)* computes the minimum quantum required by C_1 so that the end-to-end delay upper bound of all the flows (v_1 – v_5) in C_1 is less than their deadline (250 μsec). This process is shown in Table 3.3. Since the end-to-end delay for each flow in C_1 can be different we focus only on the highest of these delays, i.e. in Table 3.3 we have max $D_{C_1}^{E2E} = \max_{1 \leq i \leq 5} \{D_{v_i}^{E2E}\}$. In each step, the selection of Q_{C_1} value is based on a binary search so that $Q_{C_1} = \frac{upperValue + lowerValue}{2}$ where *upperValue* and *lowerValue* depends upon the given condition (increase Q_{C_1} or decrease Q_{C_1}). Initially, we have *upperValue* = *lowerValue* = Q = 646 bytes.

So, in the first step maximum quantum $Q_1 = 646$ bytes is assigned to C_1. In this case, we get max $D_{C_1}^{E2E} = 95.53$ μsec. Since this computed delay is less than the deadline (250 μsec), Q_1 can be reduced. It will increase the worst-case delay for C_1 flows.

In order to reduce Q_{C_1}, the *upperValue* remains same as previous one (646 bytes) but the *lowerValue* should be reduced. Since the worst-case delay computation in NC is based on the convergence of overall arrival curve and the service curve, the service rate ρ_{C_1} must be more than the long-term arrival rate $r_{C_1} = \sum_{v_i \in C_1} \frac{l_{v_i}^{max}}{T_{v_i}}$ of the cumulative flows in C_1 at each switch output port. In the given network we have $r_{C_1} = 0.056$ bits/μsec which is observed at switch S_2. So, for ρ_{C_1} to be greater than 0.056, Q_{C_1} cannot be less than 7 bytes (from equation 2.3). Thus, we have *lowerValue* = 7 bytes.

So, the next value obtained by binary search is $Q_1 = \frac{646+7}{2} \approx 326$ bytes. In this case, the maximum end-to-end delay for C_1 flows is 209.17 μsec . Since the computed delay is still less than the delay constraint on C_1, Q_1 can be further decreased. This process is repeated until the computed delay is increased to a value which is near (but smaller) to the C_1 delay constraint (250 μsec). It can be observed from Table 3.3 that maximum delay bound in C_1 is increased to 249.49 μsec when Q_{C_1} is 263 bytes. So, the value returned by *QuantumMin(C_1,Q)* is $Q_1 = 263$ bytes.

Since Q_{C_1} is less than *QResid* (646 bytes), it can be reserved for C_1, so the residual quantum for other classes is *QResid* = 646 - 263 = 383 bytes (line 13). In the algorithm, *min* is the smallest factor which is obtained from the ratio of the quantum assigned to a class and its maximum frame length. This factor will be used to improve the value of Q (line 26). From class C_1, we get $min = \frac{263}{100} = 2.63$ (line 14–16).

Next, critical class is C_2. The above mentioned steps are repeated for C_2 so that the

Table 3.3 Quantum assignment in the network (Figure 3.2).

(bytes)				(μsec)		
Q	Q_{C_1}	Q_{C_2}	Q_{C_3}	max $D_{C_1}^{E2E}$	max $D_{C_2}^{E2E}$	max $D_{C_3}^{E2E}$
646	646	-	-	95.53	-	-
	326	-	-	209.17	-	-
	166	-	-	350.44	-	-
	246	-	-	263.03	-	-
	286	-	-	233.23	-	-
	266	-	-	247.25	-	-
	256	-	-	254.89	-	-
	261	-	-	251.01	-	-
	263	-	-	249.49	-	-
		383	-		173.76	-
		195	-		305.28	-
		101	-		499.97	-
			282			302.94
640	640	-	-	95.53	-	-
	323	-	-	208.68	-	-
	165	-	-	348.4	-	-
	244	-	-	262.19	-	-
	283	-	-	232.92	-	-
	263	-	-	247.08	-	-
	253	-	-	254.81	-	-
	258	-	-	250.89	-	-
	260	-	-	249.34	-	-
		380	-		173.35	-
		193	-		305.2	-
		100	-		499.4	-
			280			301.78

minimum value of quantum obtained from *QuantumMin(C_2, Q)* is $Q_2 = 101$ bytes (also shown in Table 3.3). Since Q_{C_2} is less than $QResid$ (383 bytes), it can be reserved for C_2, and the residual quantum is reduced to $QResid = 383 - 101 = 282$ bytes. The factor min is updated to $\frac{101}{100} = 1.01$.

The quantum for all the critical classes is now computed, so the residual quantum can be assigned to the non-critical class C_3, we get $Q_{C_3} = QResid = 282$ bytes (line 22).

Next, we verify whether the obtained solution is improvable or not valid (line 25). Let us assume $\epsilon = 0$. We have $min = 1.01$ (i.e. > 1) which means that for the given distribution of Q all the classes received a quantum greater than their maximum frame

size. The quantum for a class has to be greater than or equal to the size of the largest frame in the class (equation 2.2). Hence, we can reduce Q until a valid distribution where at least one class receives quantum equal to its largest frame size is achieved. Such value of Q cannot be reduced further and hence corresponds to optimal Q. Indeed, the factor min represents the convergence towards the non improvable value of Q. If $min < 1$, then Q is lower than the optimal value and if $min > 1$, then Q is higher than the optimal value. For $Q = 646$ bytes, since $min > 1$, Q can be reduced to a value $Q = \frac{Q}{min} = \frac{646}{1.01} \approx$ 640 bytes (line 26).

The distribution of this new $Q = 640$ bytes among each class is done by repeating the whole process. The corresponding value of quantum distribution is also given in Table 3.3. The algorithm stops at $Q = 640$ with distribution $Q_{C_1} = 260$, $Q_{C_2} = 100$ and $Q_{C_3} = 280$ bytes. At this point, the factor min is reduced to 1 and the optimal solution is achieved. It can also be observed from Table 3.3 that the residual bandwidth for C_3 is higher than the previous case and the end-to-end delay bound is reduced.

3.5 Conclusion

In this chapter we have seen the importance of quantum assignment in DRR scheduler and how it can be optimised to efficiently utilise the capacity of DRR to offer guaranteed long-term bandwidth to flow classes. We have presented an algorithm to find the optimal quantum distribution which ensures that no critical class deadline will be missed and that the residual bandwidth for the non-critical class is maximum. We have illustrated, with the help of a small example, that the given approach leads to smaller worst-case delays for non-critical flows. In Chapter 5, the performance of our algorithm is evaluated through an industrial case study.

The classical NC approach used in this chapter allows the computation of worst-case delays independently for each traffic class. This is possible since this approach does not take into account the traffic from individual classes. It assumes maximum traffic which is limited by the quantum assigned to these classes in each service round. In the next chapter, we show that such an assumption can be too pessimistic and we propose an optimised NC approach to mitigate this pessimism.

Chapter 4

An Optimized NC Approach to WCTT Analysis in a Switched Ethernet Network with DRR Scheduling

Contents

4.1 Introduction 80

4.2 Classical NC approach 80

4.2.1 Modelling switched Ethernet network with DRR schedulers in classical NC approach 80

4.2.2 Pessimism in classical NC approach 82

4.3 Optimizing NC to reduce pessimism 86

4.3.1 Maximized service of interfering classes 87

4.3.2 Effective maximum load of interfering classes 90

4.3.3 Limitation of the service to the load 91

4.4 Flow scheduling at source end-node 92

4.4.1 Introducing offset 93

4.4.2 Integrating offset in NC arrival curve 96

4.5 Evaluation of optimised NC approach 101

4.6 Conclusion 102

4.1 Introduction

In this chapter, we will see that the delay computation based on classical NC approach can be pessimistic. There are two main sources of this pessimism (1) The network service modelled by the service curve does not take into account the effective traffic in the network. (2) The flow model does not take into account the temporal separation of flows at their source. The main objective of this chapter is to show how the delay computation in classical NC approach can be pessimistic and to propose an optimised NC approach to compute tight delay bounds.

The organisation of this chapter is as follows. We briefly recall the classical NC approach in Section 4.2 and show the sources of pessimism in this approach. In Section 4.3 we propose an optimized NC approach for DRR scheduler based real-time switched Ethernet networks. In section 4.4 we integrate the scheduling of flows at their end-nodes in the NC flow model. In Section 4.5 we compare the delay computed by classical NC approach and optimised NC approach. Section 4.6 concludes the chapter.

4.2 Classical NC approach

The NC approach used until now in this thesis, which was described in section 2.4, was proposed by Boyer et al. in [BSS12]. It allows the computation of worst-case delay bounds for each traffic class independently. This is possible because it does not take into account the traffic of individual classes, only the knowledge of the sum of quantum is needed. Such computation may involve significant pessimism, depending on the actual traffic in each class. In the following paragraphs, we briefly recall this approach and then we show how the computation can be pessimistic.

4.2.1 Modelling switched Ethernet network with DRR schedulers in classical NC approach

The classical NC approach models the flows at source end-node by an arrival curve which represents an over-estimation of the traffic of this flow at any instant t as:

$$\alpha(t) = \frac{l^{max}}{T}t + l^{max}, \forall\ t > 0$$

where the l^{max} is maximum frame length and T is the minimum inter-frame arrival time.

The flows arriving at a switch output port h controlled by a DRR scheduler are differentiated based on predefined classes. All the flow frames of a class C_x are stored in

the same FIFO queue. Thus, the overall arrival curve used to constrain the traffic of C_x flows is the cumulative curve obtained by the sum of arrival curves of all the flows of C_x at h:

$$\alpha^h_{C_x} = \sum_{i \in \mathcal{F}^h_{C_x}} \alpha^h_i(t) \tag{4.1}$$

where $\mathcal{F}^h_{C_x}$ is the set of C_x flows at output port h.

Since a flow frame can be delayed by other frames between its source node e_x and a switch port h, a jitter J^h has to be introduced. It is the difference between the worst-case delay and the best-case delay for a frame of this flow from its source node to h. The integration of this jitter in the arrival curve is explained in section 1.5.2.

The service provided to C_x flows at an output port h with link rate R is modelled by a service curve as:

$$\beta^h_{C_x}(t) = \rho^h_{C_x}[t - sl - \Theta^h_{C_x}]^+ \tag{4.2}$$

where, sl is the switching latency and $[a]^+ = \max\{0, a\}$.

This service is minimised by considering maximum interference from competing classes that leads to the maximum scheduler latency $\Theta^h_{C_x}$ at beginning of C_x service and the minimum service rate $\rho^h_{C_x}$ (theoretical service rate) for the rest of the active period of C_x. Since, in DRR scheduler, the bandwidth R is shared by all the classes at the given output port, each class C_x receives a fraction $\rho^h_{C_x}$ of R. This fraction is based on its assigned quantum $Q^h_{C_x}$:

$$\rho^h_{C_x} = \frac{Q^h_{C_x}}{\sum\limits_{1 \le j \le n} Q_{C_j}} \times R \tag{4.3}$$

where, n is the number of classes at h.

The maximum scheduler latency for C_x is when it is delayed (by a value X_{C_x}) by all the competing classes at h before being served for the first time and this first service is a reduced service (i.e. less than theoretical service rate). The scheduler latency is given by,

$$\Theta^h_{C_x} = X^h_{C_x} + Y^h_{C_x} \tag{4.4}$$

where,

$$X^h_{C_x} = \frac{\sum\limits_{j=1, j \neq x}^{n} (Q^h_{C_j} + \Delta^h_{C_j})}{R}$$

and

$$Y_{C_x}^h = \frac{(Q_{C_x}^h - \Delta_{C_x}^{max,h}) + \sum_{j=1,j\neq x}^{n} Q_{C_j}^h}{R} - \frac{(Q_{C_x}^h - \Delta_{C_x}^{max,h})}{\rho_{C_x}^h}$$

where, $\Delta_{C_x}^{max,h}$ (= $l_{C_x}^{max,h}$ - 1) is the maximum deficit in C_x.

The worst-case delay experienced by a flow v_i of C_x at h is upper bounded by the maximum horizontal difference between the cumulative arrival curve $\alpha_{C_x}^h$ and the service curve $\beta_{C_x}^h$ offered to C_x. This difference is given by :

$$D_i^h = \sup_{s\geq 0}(\inf\{\tau \geq 0 | \alpha_{C_x}^h(s) \leq \beta_{C_x}^h(s+\tau)\})$$

The worst-case end-to-end delay in the network is the sum of worst-case delays at each port in the path $\mathcal{P}_i$ of v_i:

$$D_i^{E2E} = \sum_{h\in\mathcal{P}_i} D_i^h$$

4.2.2 Pessimism in classical NC approach

The classical NC approach models the service for an active class C_x by the service curve $\beta_{C_x}^h$ which assumes a scheduler latency $\Theta_{C_x}^h$ at the beginning of service and then a minimum service rate $\rho_{C_x}^h$ throughout the active period of C_x. The computation of $\Theta_{C_x}^h$ and $\rho_{C_x}^h$ is based on an assumption that in each DRR service round rd_k, during the active period of C_x, the competing classes C_y ($y \neq x$) are always active and each C_y transmits frames of at least the size of its quantum value $Q_{C_y}^h$. Such an assumption might be pessimistic as it does not take into account the exact composition of the traffic in these classes. Indeed, the traffic from one or several C_y classes might be too low to consume the credit equal to its quantum values $Q_{C_y}^h$ in each round.

Let us consider a very basic scenario where 2 classes (C_1 and C_2) are served by a DRR scheduler. They both are assigned quantum of 200 bytes. If C_1 has 10 frames of 100 bytes to be served then it will take 5 rounds to serve all the frames as it will consume its quantum of 200 bytes in the transmission of 2 frames per round. During the service of C_1, if the competing class C_2 has only one frame of 100 bytes to be served then C_2 cannot delay C_1 frames in more than one DRR round. In such a scenario, the computation based on the classical NC approach will be clearly pessimistic.

Let us illustrate this pessimism with the small network shown in Figure 4.1. The flow specifications are given in Table 4.1. The link rate is $R = 100$ bits/μsec and the flow differentiation is such that: v_1 to v_5 belongs to C_1, v_6 to v_{10} belongs to C_2 and v_{11} to

v_{14} belongs to C_3. The switch output ports are managed by DRR scheduler where each class is assigned equal quantum of 199 bytes.

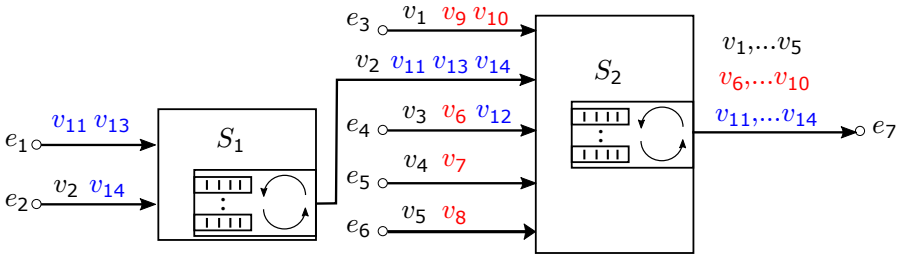


Figure 4.1 A switched Ethernet network with DRR scheduling

Table 4.1 Flow specifications

Flows (v_i)	Minimum inter-frame time (T_i)	Frame lengths (l_i)
v_6, v_9, v_{10}, v_{13}	128	80–100
v_1, v_7, v_8	128	80–99
v_2, v_4, v_5, v_{14}	64	80–100
v_3, v_{11}, v_{12}	64	80–99

Let us focus on switch S_2 to analyse the service provided to C_1 flows. One possible scenario of service provided to C_1 flows is shown in Figure 4.2. The service rounds in Figure 4.2 are based on the classical approach that focuses on achieving maximised scheduler latency and minimum service rate for the class being analysed (i.e. C_1).

In Figure 4.2, C_1 becomes active at t_1 where it just misses its opportunity to be served. In round rd_1, C_1 experiences maximum delay when C_2 and C_3 consume their maximum credit of $Q_{C_y}^{S_2} + \Delta_{C_y}^{max,S_2} = 199 + 99 = 298$ bytes ($y = 2,3$) in transmission of frames from v_6, v_7, v_8, v_{14}, v_{11} and v_{12}. This gives an initial delay of $t_1' - t_1 = 47.68$ μsec for C_1 flows. No deficit is left for C_2 and C_3.

C_1 gets its first service at t_1' where it is assigned a credit equal to its quantum value $Q_{C_1}^{S_2} = 199$ bytes, but, it consumes only $Q_{C_1}^{S_2} - \Delta_{C_1}^{max,S_2} = 199 - 99 = 100$ bytes (i.e. minimum credit that can be consumed by an active class) in transmission of v_5 frame leaving a deficit of 99 bytes.

The round rd_2 starts at t_2. C_2 receives a credit equal to its quantum value $Q_{C_2}^{S_2} = 199$ bytes. Since there is only one frame (from v_{10}) of size 100 bytes in C_2 buffer, this frame is transmitted and the deficit is reset to 0. At this point, since all frames in C_3 are already served in the previous round and there are no new frames in C_3 buffer, the

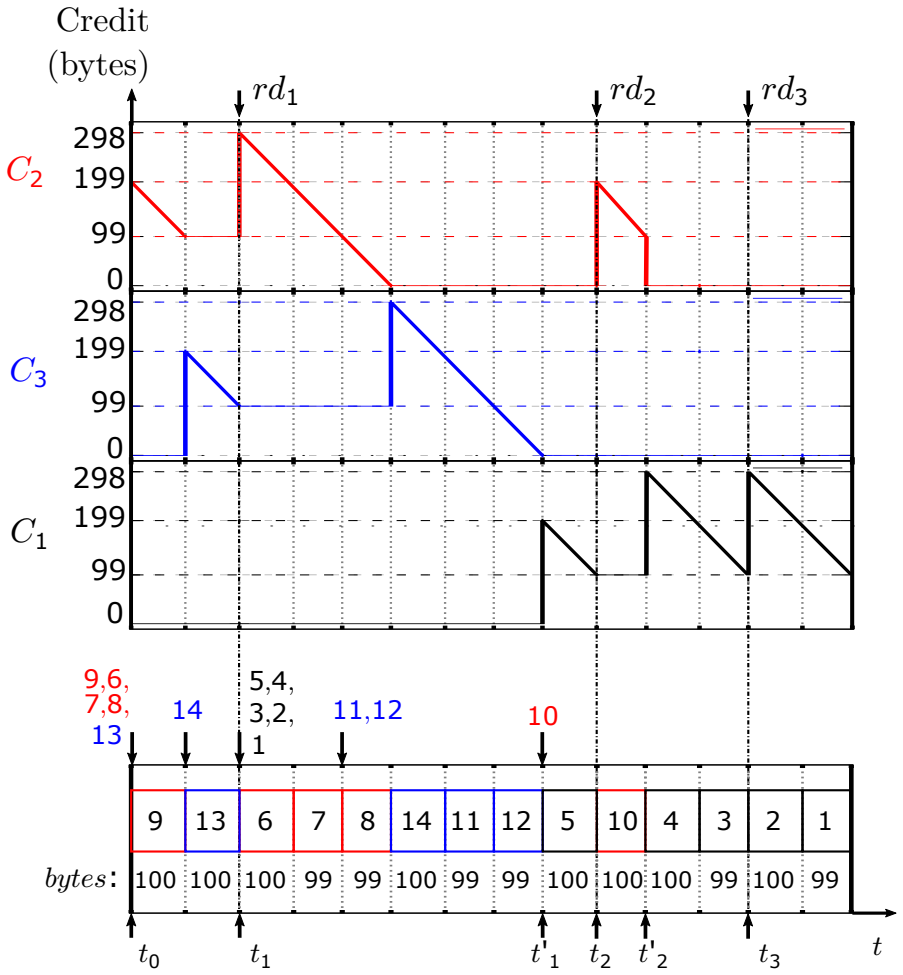


Figure 4.2 DRR scheduling rounds at output port (Figure 4.1)

scheduler moves to the service of C_1. C_1 gets a credit equal to the sum of its quantum and the deficit from previous round, i.e. $199 + 99 = 298$ bytes, and serves the frames from v_4 and v_3 leaving a deficit of 99 bytes which is less than the size of the next frame (from v_2).

The round rd_3 starts at t_3. Since C_2 and C_3 are no more active, C_1 receives its next credit of $199 + 99 = 298$ bytes and transmits the remaining frames from v_2 and v_1.

Let us now compute the scheduler latency and service rate for C_1 at S_2 in the given scenario and compare it with those considered by classical NC approach.

In classical NC approach, the scheduler latency (equation 4.4) is the sum of $X_{C_1}^{S_2}$ (delay experienced by C_1 before its first service) and $Y_{C_1}^{S_2}$ (delay due to reduced service rate in duration between beginning of first and second service). In the given scenario the

delay before the first service is $t'_1 - t_1 = 47.68$ μsec, but, in the duration between the beginning of the first service of C_1 (at t'_1) and the beginning of its second service (at t'_2), total 200 bytes (100 bytes from C_1 and 100 bytes from C_2) are served. So, the service rate received by C_1 is:

$$\frac{100}{200} \times 100 = 50 \ bits/\mu sec$$

which is greater than its theoretical service rate $\rho_{C_1}^{S_2} = 33.33$ bits/μsec (equation 4.3). After t'_2, since C_1 is the only active class, the average service rate received by C_1 flows in each scheduling round is equal to the maximum link capacity (100 bits/μsec). Thus in this scenario, C_1 does not receive any reduced service. So, the total scheduler latency for C_1 flows before being served at a service rate greater than or equal to the theoretical service rate is 47.68 μsec. Whereas, the scheduler latency considered by classical NC approach is

$$\begin{aligned}
X_{C_1}^{S_2} &= \frac{(Q_{C_2}^{S_2} + \Delta_{C_2}^{S_2}) + (Q_{C_3}^{S_2} + \Delta_{C_3}^{S_2})}{R} \\
&= \frac{(199 + 99 + 199 + 99) \times 8}{100} = 47.68 \mu sec \\
Y_{C_1}^{S_2} &= \frac{(Q_{C_1}^{S_2} - \Delta_{C_1}^{max,S_2}) + Q_{C_2}^{S_2} + Q_{C_3}^{S_2}}{R} - \frac{(Q_{C_1}^{S_2} - \Delta_{C_1}^{max,S_2})}{\rho_{C_1}^{S_2}} \\
&= \frac{((199 - 99) + 199 + 199) \times 8}{100} - \frac{(199 - 99) \times 8}{33.33} = 15.84 \mu sec \\
\Theta_{C_1}^{S_2} &= X_{C_1}^{S_2} + Y_{C_1}^{S_2} = 47.68 + 15.84 = 63.52 \mu sec
\end{aligned}$$

Figure 4.3 shows a comparison between the service received by C_1 in the given scenario and the pessimistic service ($\beta_{C_1}^{S_2} = \rho_{C_1}^{S_2}[t - \Theta_{C_1}^{S_2}]^+ = 33.33[t - 63.52]^+$) considered by classical NC approach.

For the given network, the worst-case delay computed for flow v_1 by classical NC approach is $D_1^{S_2} = 183.07$ μsec. If we compute the delay for v_1 based on the frame transmission given in Figure 4.2, we get

$$\frac{(100 + 99 + 99 + 100 + 99 + 99 + 100 + 100 + 100 + 99 + 100 + 99) \times 8}{100} = 95.52 \mu sec$$

which is much smaller than the computed worst-case delay. However, the scenario in Figure 4.2 may not be the worst-case one. But, in any case in the given network, the traffic from C_2 and C_3 is not enough to consume the credit equal to their respective quantum values in each round and hence the actual delay for v_1 is certainly much less than the computed worst-case delay.

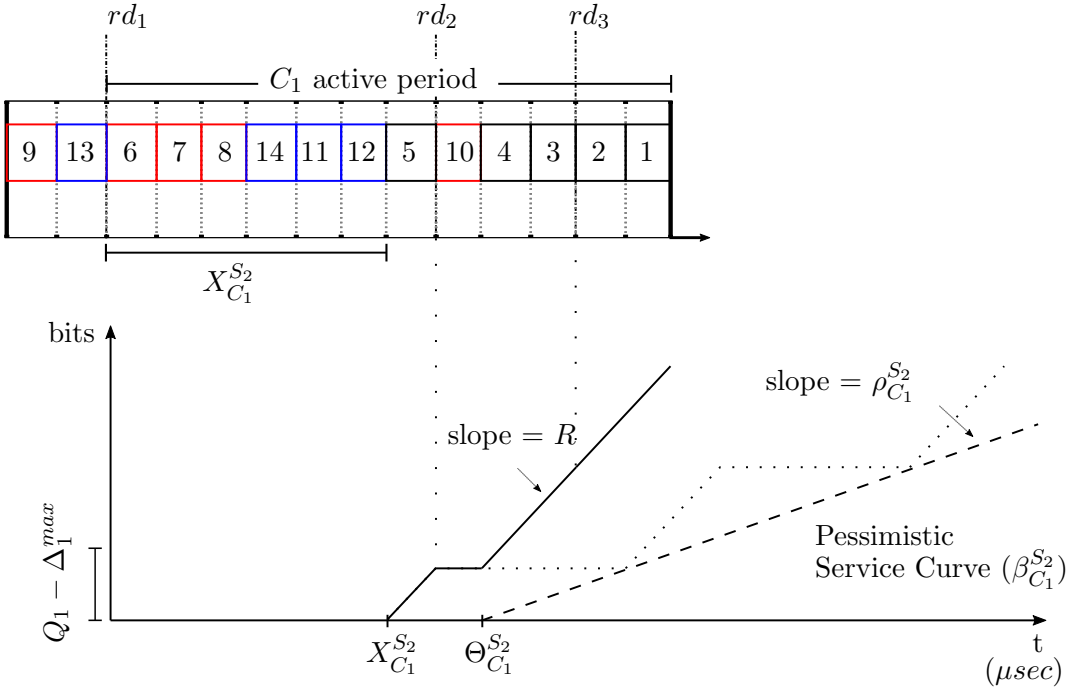


Figure 4.3 Pessimism in NC service curve

4.3 Optimizing NC to reduce pessimism

In classical NC approach, as seen in the previous section, the service curve ($\beta_{C_x}^h$) used in worst-case delay computation can be pessimistic as it does not take into account the actual traffic from each class. Actually, in order to consider the worst-case scenario for C_x flows this service curve aims to maximise the traffic from competing classes C_y ($y \neq x$). Which means at any instant (t_i) the traffic served from C_y at h cannot be higher than what is considered by $\beta_{C_x}^h$. However, the actual traffic queued in C_y buffer may or may not be less than what is considered by $\beta_{C_x}^h$, depending on the composition of C_y.

In the following paragraphs, we will see that the actual traffic in C_y at instant t_i can be upper bounded as $L_{C_y}^{max,h}(t_i)$. If this upper bound is less than the maximised traffic of C_y considered by $\beta_{C_x}^h$, then, the difference between the two represents the over-estimated portion of C_y traffic in $\beta_{C_x}^h$.

We propose an optimised NC approach that eliminates the over-estimated traffic considered by classical NC approach in order to compute tight delay bound. Let us illustrate the basic idea of optimised NC approach with the help of the small network shown in Figure 4.1. In this example, the optimization of delay computation for v_1 at S_2 proceeds as follows:

- Compute the worst-case delay for flow v_1 (from class C_1) at S_2 using classical NC approach, which is $D_1^{S_2} = 183.07$ μsec.
- Since, $D_1^{S_2}$ is the sure delay upper bound for v_1, only the traffic from competing class (C_2) that is present in C_2 buffer before this instant can participate in delaying v_1. So, compute the maximised traffic of C_2 considered by $\beta_{C_1}^{S_2}$ in interval $[0, D_1^{S_2}]$. We define this maximum traffic as *service load* $SL_{C_2}^{S_2}(D_1^{S_2})$ in Section 4.3.1.
- Compute the upper bound on actual traffic in C_2 in interval $[0, D_1^{S_2}]$. We define this upper bound as *effective maximum load* $L_{C_2}^{max,S_2}(D_1^{S_2})$ in Section 4.3.2.
- If the service load $SL_{C_2}^{S_2}(D_1^{S_2})$ is greater than the effective maximum load $L_{C_2}^{max,S_2}(D_1^{S_2})$, then the difference between the two represents the over-estimated portion of C_2 traffic in $\beta_{C_1}^{S_2}$. Otherwise, the difference is assumed to be 0.
- We repeat the same process for all the other competing classes (only C_3 in the given example).
- Finally, the delay considered in transmission of the overestimated traffic of $(SL_{C_2}^{S_2}(D_1^{S_2}) - L_{C_2}^{max,S_2}(D_1^{S_2}))$ bytes and $(SL_{C_3}^{S_2}(D_1^{S_2}) - L_{C_3}^{max,S_2}(D_1^{S_2}))$ bytes at link rate ($R = 100$ bits/μsec) can be removed from the delay upper bound $D_1^{S_2}$ in order to get the optimised delay $D_{1,opt}^{S_2}$.

4.3.1 Maximized service of interfering classes

The maximised service of a competing class C_y, considered by a service curve $\beta_{C_x}^h$ of a class C_x (equation (4.2)), is not same for all the time intervals.

In order to compute the maximised service of competing classes C_2 and C_3, let us analyse the different intervals considered in $\beta_{C_1}^{S_2}$ (Figure 4.4).

- The service curve starts at $t = 0$ when the first frame in C_1 queue arrives at S_2. The first interval [0, $X_{C_1}^{S_2}$] corresponds to the delay $X_{C_1}^{S_2} = 47.68$ μsec experienced by C_1 flows before being served for the first time. In this interval, the classical NC approach assumes that C_2 and C_3 get the service of maximum possible credit ($Q_{C_2}^{S_2} + \Delta_{C_2}^{max,S_2} = 199 + 99 = 298$ bytes and $Q_{C_3}^{S_2} + \Delta_{C_3}^{max,S_2} = 199 + 99 = 298$ bytes).
- At $X_{C_1}^{S_2}$, C_1 receives the first service of minimum credit ($Q_{C_1}^{S_2} - \Delta_{C_1}^{max,S_2} = 199 - 99 = 100$ bytes). Next interval ($X_{C_1}^{S_2}$, t_N] corresponds to the duration between first and second service of C_1 flows. In this interval, C_2 and C_3 receive service of credit equal to their quantum values ($Q_{C_2}^{S_2} = 199$ bytes and $Q_{C_3}^{S_2} = 199$ bytes).

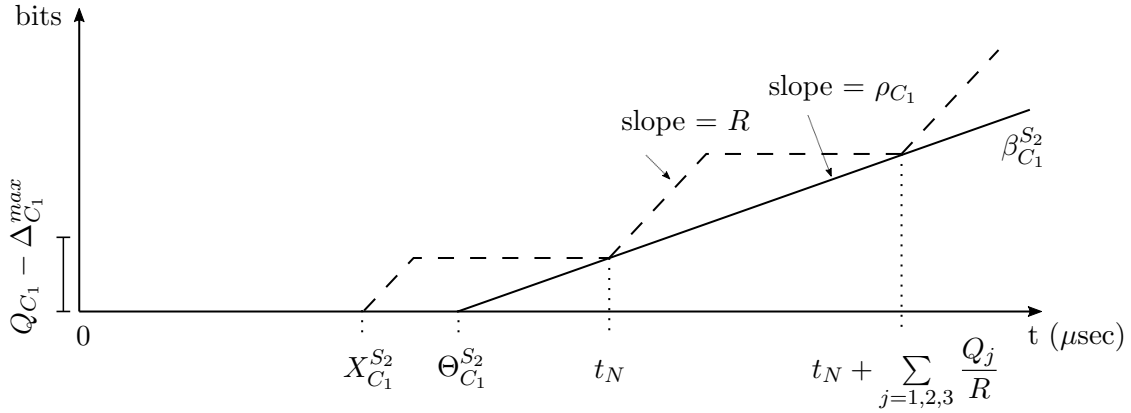


Figure 4.4 Pessimistic Service Curve of C_1 at S_2 (Figure 4.1)

- The following intervals are all identical. In these intervals, each class receives service of credit equal to its quantum value.

These intervals in service curve of classical NC approach can be generalised for a DRR scheduler serving any number of classes (n^h). For a class C_x with service curve $\beta_{C_x}^h$, we define a function called *service load* $SL_{C_y}^h$(t) to compute the maximized traffic in competing class C_y as:

$$SL_{C_y}^h(t) = \begin{cases} 0, & t < X_{C_x}^h \\ Q_{C_y}^h + \Delta_{C_y}^{max,h}, & X_{C_x}^h \leq t < t_N \\ Q_{C_y}^h + \Delta_{C_y}^{max,h} + \left(1 + \left\lfloor \dfrac{R \times (t - t_N)}{\sum_{j=1}^{n^h} Q_j^h} \right\rfloor\right) Q_{C_y}^h, & t_N \leq t \end{cases} \tag{4.5}$$

where,

$$t_N = X_{C_x}^h + \frac{1}{R}\left(Q_{C_x}^h - \Delta_{C_x}^{max,h} + \sum_{j=1, j\neq x}^{n^h} Q_j^h\right)$$

In service load $SL_{C_y}^h$(t), the load corresponding to a given interval is taken into account only at the end of this interval. For instance, since the competing class C_y is served $Q_{C_y}^h + \Delta_{C_y}^h$ bytes in interval $[0, X_{C_x}^h]$, the load of C_y before $X_{C_x}^h$ is 0. The $Q_{C_y}^h + \Delta_{C_y}^h$ bytes from C_y considered at $t = X_{C_x}^h$ remains unchanged until the next service of C_y. Then, C_y is served Q_y^h bytes in interval $(X_{C_x}^h, t_N]$ which is also considered at $t = t_N$. In the following intervals, the load of Q_y^h bytes is accumulated at the end of each interval.

Thus, $SL^h_{C_y}(\mathrm{t})$ is a lower-bound on the maximized service considered by $\beta^h_{C_x}$. This service load is also shown in Figure 4.5.

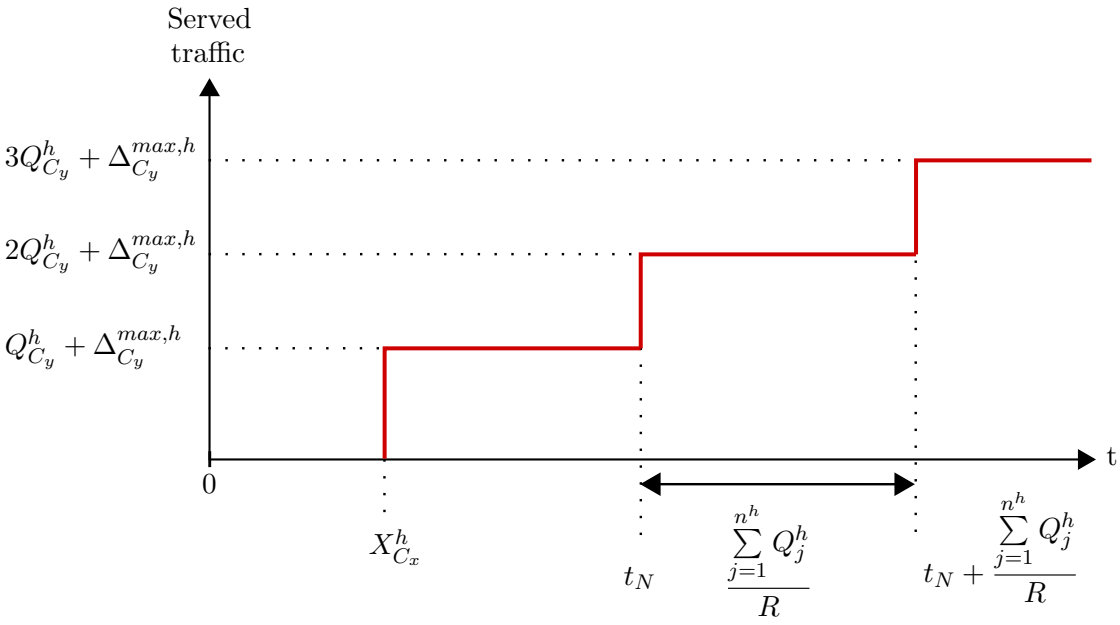


Figure 4.5 Lower bound on maximum service load in DRR scheduler

For v_1, the service load from competing classes C_2 and C_3 in the interval $[0, D_1^{S_2}]$ can be computed as:

We have,

$$
\begin{aligned}
D_1^{S_2} &= 183.07\mu sec \\
X_{C_1}^{S_2} &= 47.68\mu sec \\
t_N &= X_{C_1}^{S_2} + \frac{1}{R}\left(Q_{C_1}^{S_2} - \Delta_{C_1}^{S_2} + Q_{C_2}^{S_2} + Q_{C_3}^{S_2}\right) \\
&= 47.68 + \frac{1}{100}\left((199 - 99 + 199 + 199) \times 8\right) = 87.52\mu sec
\end{aligned}
$$

Since, $D_1^{S_2} > t_N$, *therefore*

$$\begin{aligned} SL_{C_2}^{S_2}(D_1^{S_2}) &= Q_{C_2}^{S_2} + \Delta_{C_2}^{max,S_2} + \left(1 + \left\lfloor \frac{R \times (D_1^{S_2} - t_N)}{\sum_{j=1}^{3} Q_j^{S_2}} \right\rfloor\right) Q_{C_2}^{S_2} \\ &= (199+99) \times 8 + \left(1 + \left\lfloor \frac{100 \times (183.07 - 87.52)}{(199+199+199) \times 8} \right\rfloor\right)(199 \times 8) \\ &= 7160\ bits = 895\ bytes \end{aligned}$$

Similarly, we get $SL_{C_3}^{S_2} = 895$ bytes.

4.3.2 Effective maximum load of interfering classes

In classical NC approach, the flow arrival in a class buffer at an output port is constrained by an overall arrival curve (equation (4.1)), which means, an overall arrival curve $\alpha_{C_y}^h(\mathrm{t})$ represents the upper bound on traffic from C_y flows at all time t.

Thus, the upper bound on C_2 and C_3 traffic in interval $[0, D_{C_1}^{S_2}]$ can be given by $\alpha_{C_2}^{S_2}(D_{C_1}^{S_2})$ and $\alpha_{C_3}^{S_2}(D_{C_1}^{S_2})$. We call such upper bound as an *effective load* ($L_{C_y}^{max,S_2}(D_{C_1}^{S_2})$) of C_y (y = 2,3).

At S_2, the traffic in C_2 is composed of 5 flows: v_6 from e_4, v_7 from e_5, v_8 from e_6, and v_9 and v_{10} from e_3 (see Figure 4.1). Based on equation (4.1) the overall arrival $\alpha_{C_2}^{S_2}(\mathrm{t})$ can be obtained by the sum of individual arrival curves of each flow in C_2. However, since the flows v_9 and v_{10} arrives at S_2 from same input link, they are serialized and can be represented by a cumulative curve with burst limited to the size of maximum burst in arrival curves of v_9 and v_{10} and the arrival rate limited by link rate (R). This serialization effect is explained in Section 1.5.2. We have,

$$\alpha_{C_2}^{S_2}(t) = \alpha_{v_6}^{S_2} + \alpha_{v_7}^{S_2} + \alpha_{v_8}^{S_2} + \min\{((R \times t) + \max\{b_{v_9}^{S_2}, b_{v_{10}}^{S_2}\}), (\alpha_{v_9}^{S_2} + \alpha_{v_{10}}^{S_2})\}$$

where, $\alpha_{v_6}^{S_2} = (0.00625 \times t) + 800.1$, $\alpha_{v_7}^{S_2} = (0.00618 \times t) + 792.05$, $\alpha_{v_8}^{S_2} = (0.00618 \times t) + 792.05$, $\alpha_{v_9}^{S_2} = (0.00625 \times t) + 800.1$, and $\alpha_{v_{10}}^{S_2} = (0.00625 \times t) + 800.1$

Therefore, the effective load of C_2 in interval $[0, D_{C_1}^{S_2}]$ is

$$\begin{aligned} L_{C_2}^{S_2}(D_{C_1}^{S_2}) &= \alpha_{C_2}^{S_2}(D_{C_1}^{S_2}) \\ &= \alpha_{C_2}^{S_2}(183.07) \\ &= 3990\ bits = 498.75\ bytes \end{aligned} \tag{4.6}$$

Similarly, we get $L_{C_3}^{S_2}(D_{C_1}^{S_2}) = 399$ bytes. The overall arrival curves of C_2 and C_3 are also shown in Figure 4.6.

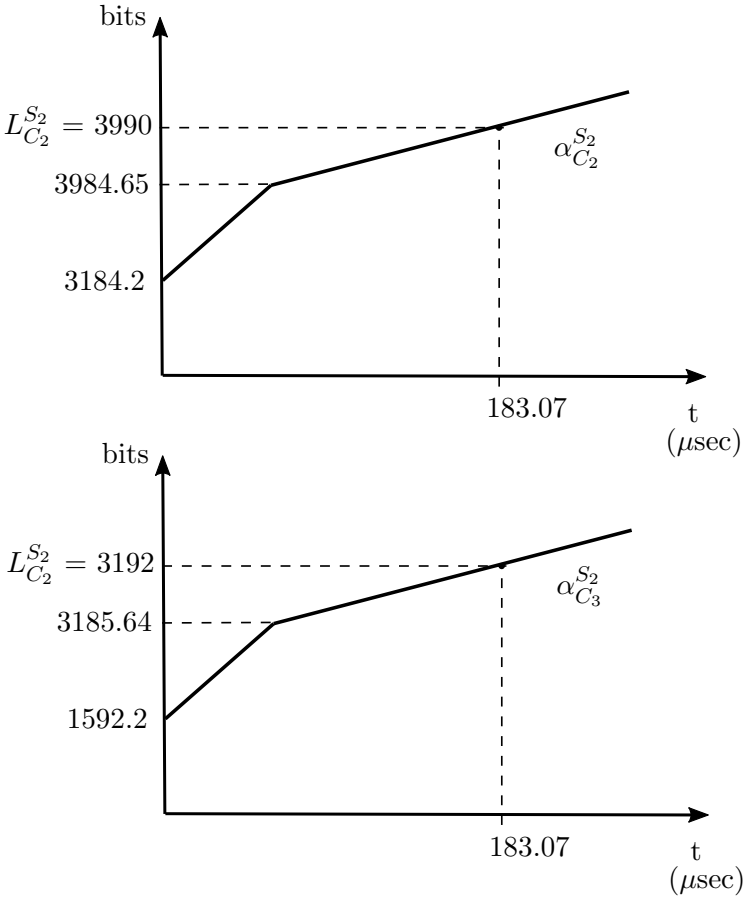


Figure 4.6 Effective load of C_2 and C_3 flows

4.3.3 Limitation of the service to the load

In the duration $[0, D_1^{S_2}]$, if the service load $(SL_{C_y}^{S_2}(D_1^{S_2}))$ taken into account by the classical NC service curve $(\beta_{C_1}^{S_2})$ is more than the effective maximum load $(L_{C_y}^{max,S_2}(D_1^{S_2}))$, then the difference between the two represents the pessimistic over-estimation of traffic from C_y $(y = 2, 3)$ considered in the computation of delay $D_1^{S_2}$. This means that $D_1^{S_2}$ also includes the transmission time of this over-estimated traffic. Since this over-estimated portion of C_y $(y = 2, 3)$ is also assumed to be transmitted at link rate R, thus, to reduce

the pessimism in computed delay ($D_1^{S_2}$), we can remove the following value :

$$\frac{\max\{SL^h_{C_2}(D_1^{S_2}) - L^{max,S_2}_{C_2}(D_1^{S_2}), 0\}}{R} + \frac{\max\{SL^h_{C_3}(D_1^{S_2}) - L^{max,S_2}_{C_3}(D_1^{S_2}), 0\}}{R}$$
$$= \frac{(895 - 498.75) \times 8}{100} + \frac{(895 - 399) \times 8}{100}$$
$$= 31.70 + 39.68 = 71.38\ \mu sec$$

Therefore, the optimised delay is

$$D^{S_2}_{1,opt} = D_1^{S_2} - 71.38 = 183.07 - 71.38 = 111.69 \mu sec$$

This step can be generalised for a DRR scheduler serving any number of classes (n^h). If the delay computed by classical NC approach for a flow v_i of class C_x is D_i^h, then the optimized delay v_i is given by:

$$D^h_{i,opt} = D^h_i - \frac{\sum_{y=1, y\neq x}^{n^h} \max\{SL^h_{C_y}(D^h_i) - L^{max,h}_{C_y}(D^h_i), 0\}}{R} \tag{4.7}$$

4.4 Flow scheduling at source end-node

The optimisation proposed in the previous section focuses on improving the service model in classical NC approach. In this section, we propose some improvements in flow model of classical NC approach in order to compute tighter delay bounds.

The classical NC approach models the flow arrival at an output port h as an overall arrival curve $\alpha^h_{C_x}$ (equation 4.1) which is the sum of the arrival curves of all the inputs to h. This sum represents the maximum traffic in C_x from all the inputs links. However, this arrival curve does not make any assumption on generation instant of frames at their source end-node, it simply considers that frames of any two different flows can be generated at the same time, which may not be the case in some situations.

In a real-time switched Ethernet network that does not utilise any global synchronisation clock, like the AFDX network, the flow emission at each end-node is independent. In such networks, the flow frames can be generated at the same time at different end-nodes so for these flows the arrival curve considered by classical NC approach is realistic. However, for the flows generated at same end-node, this is not always the case. Indeed, the periodic flows emitted by the same end-node are scheduled based on a local clock. These flows are called locally synchronized and they are dependent. The scheduling at the end-node

ensures a minimum temporal separation of frames from different flows. Thus, the frames of these flows cannot be generated at the same time.

In the following paragraphs, we will see that such separation of flows reduces the effective traffic in the network.

4.4.1 Introducing offset

Let us illustrate the impact of temporal separation of flows in the network example considered in the previous section. This network is recalled in Figure 4.7. This network supports FIFO scheduling at end-nodes and DRR scheduling at switch output ports. The transmission rate is $R = 100$ bits/μsec and the switching latency is $sl = 8$ μsec.

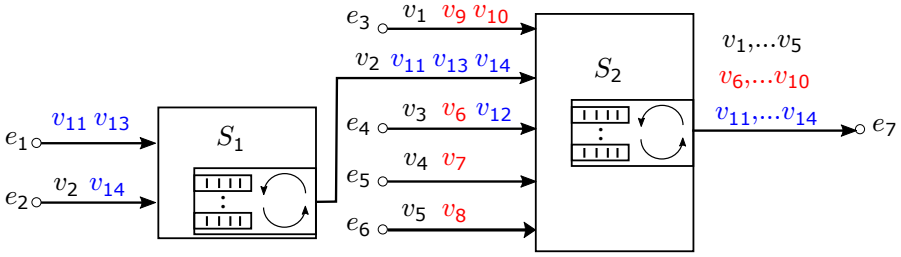


Figure 4.7 A recall of switched Ethernet network in Figure 4.1

Let us focus on the end-to-end transmission of flow v_{11} in its path e_1–S_1–S_2–e_7. The temporal separation between flows is characterised by an offset which constraints the arrival of flows at output ports.

At the source e_1, the release time of first frame of each flow is fixed by *definite offset*. Let us assume, the emission of first frames of v_{11} and v_{13} are separated by 32000 μsec, i.e. they have definite offset $O^{e_1}_{d,v_{11}} = 0$ and $O^{e_1}_{d,v_{13}} = 32000$ μsec. The following frames of these flows are emitted based on their minimum inter-frame arrival time, $T_{v_{11}} = 64000$ μsec and $T_{v_{13}} = 128000$ μsec (from Table 4.1). Figure 4.8 shows the comparison of frame sequences of v_{11} and v_{13} with and without temporal separation, where $\uparrow^{v_i}$ represents the arrival instant of v_i frame in output port buffer. In the upper part of Figure 4.8, there is no separation between the flows and the frames arriving at the output port at the same time are delayed by one another before being transmitted over the output link. Whereas, in the presence of an offset, the frames are transmitted as soon as they arrive at the output port. The frames of v_{11} are ready at time instant $k \times T_{v_{11}} = k \times 64000$ μsec while frames of v_{13} are ready at time instant $32000 + (m \times T_{v_{13}}) = 32000 + (m \times 128000)$ μsec, where $k = 1, 2, \ldots$ and $m = 0, 1, 2, \ldots$.

As these frames propagate in the network, they may experience different delays at

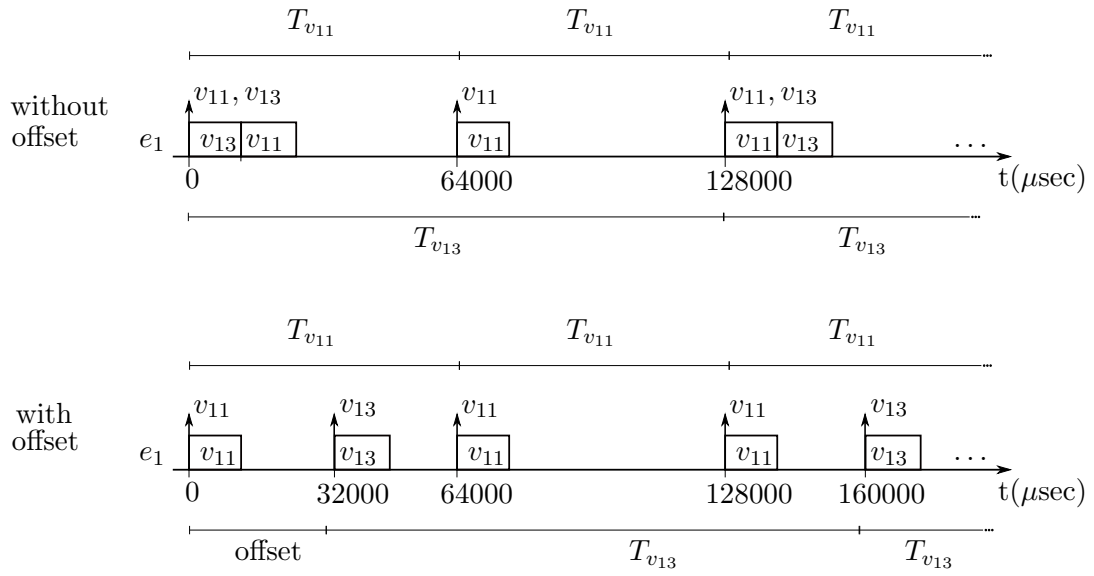


Figure 4.8 Frame sequences at e_1 with and without temporal separation of v_{11} and v_{13}

each output port depending upon their size and waiting time in the buffer. This difference in delays affects the separation of flows. For instance, the maximum size frame of v_{11} ($99 \times 8 = 792$ bits) is transmitted in $\frac{792}{100} = 7.92$ μsec, and the frame of v_{13} with maximum size $100 \times 8 = 800$ bits is transmitted in 8 μsec. Since its emission at e_1, v_1 arrives in next output port S_1 in $7.92 + sl = 7.92 + 8 = 15.92$ μsec and v_2 arrives in $8 + sl = 8 + 8 = 16$ μsec. Thus, the separation time in arrival of consecutive frames of v_{11} and v_{13} at S_1 is increased to $32000 + 16 - 15.92 = 32000.08$ μsec and it is decreased to $64000 + 15.92 - 32000 - 16 = 31999.92$ μsec in reverse order (see Figure 4.9).

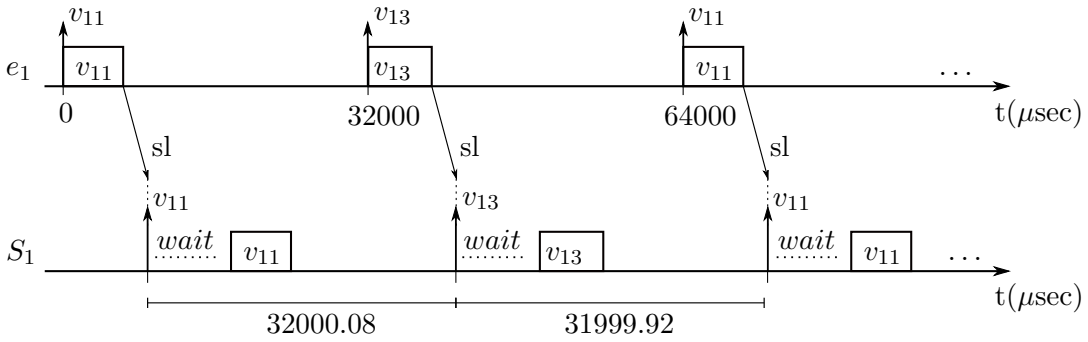


Figure 4.9 v_{11} and v_{13} frame sequence at S_1

As these frames compete with other flow frames (v_2 and v_{14}) they may experience some waiting in S_1 buffer, which further affects their temporal separation.

At any output port h, the minimum separation between the two frames emitted from the same source e_x can be defined by *relative offset*. The computation of relative offset has to take into account flow jitters. For each computation of a relative offset, a given flow has to be considered as a reference (a.k.a. benchmark flow v_b). For frames f_b and f_i from flows v_b and v_i respectively starting from same end-node e_x, the relative offset $O^{e_x}_{r,b,i}$ at e_x is based on their definite offset and minimum inter-frame time. On the path from e_x till h, the separation between f_b and f_i is reduced to minimum when f_b experiences its maximum delay $D^{e_x,h}_{max,b}$ and f_i experiences its minimum delay $D^{e_x,h}_{min,i}$. Thus, the relative offset at h is given by:

$$O^{h}_{r,b,i} = O^{e_i}_{r,b,i} - (D^{e_x,h}_{max,b} - D^{e_x,h}_{min,i}) \tag{4.8}$$

Since f_b and f_i share the same input link of h, they are serialized and, hence, the relative offset cannot be less than the transmission time (tr_{v_b}) of f_b. We have:

$$O^{h}_{r,b,i} = \max\{O^{h}_{r,b,i}, tr_{v_b}\} \tag{4.9}$$

When v_{11} is considered as benchmark flow, the relative offset between v_{11} and v_{13} at S_2 is:

$$O^{S_2}_{r,11,13} = O^{e_1}_{r,11,13} - (D^{e_1,S_2}_{max,11} - D^{e_1,S_2}_{min,13})$$

The minimum delay $D^{e_1,S_2}_{min,13}$ in v_{13} before its arrival at S_2 corresponds to the scenario where it is not delayed by any other flow. Thus, this delay is only due to the scheduler latency and the transmission time at each output port in the path of v_{13}, we have $D^{e_1,S_2}_{min,13}$ = $(tr^{e_1}_{v_{13}})$ + $(sl + tr^{S_1}_{v_{13}})$ = 8 + 8 + 8 = 24 μsec. The maximum delay $D^{e_1,S_2}_{max,11}$ in v_{11} before its arrival at S_2 is the sum of worst-case delays at each output port in the path of v_{11}. This maximum delay can be computed with the help of NC approach however the classical NC approach does not take into account the temporal separation of flows thus, in the next section, we propose a method to integrate this temporal separation of flows in arrival curves. We will see that the maximum delay $D^{e_1,S_2}_{max,11}$ is $D^{e_1}_{11}$ + $D^{S_1}_{11}$ = 7.92 + 39.84 = 47.76 μsec. Therefore, the relative offset between v_{11} and v_{13} is $O^{S_2}_{r,11,13}$ = 32000 - (47.76 - 24) = 31976.24 μsec. Figure 4.10 shows the relative offsets between v_{11} and v_{13} while considering each flow as benchmark flow.

It is worth noting that in DRR scheduling the information relative to offset between flows of different classes exists but will not be considered since classes are considered independently. Therefore, the computation of relative offset is limited to individual class. In the next section, we show how the offset between the flows can be taken into account

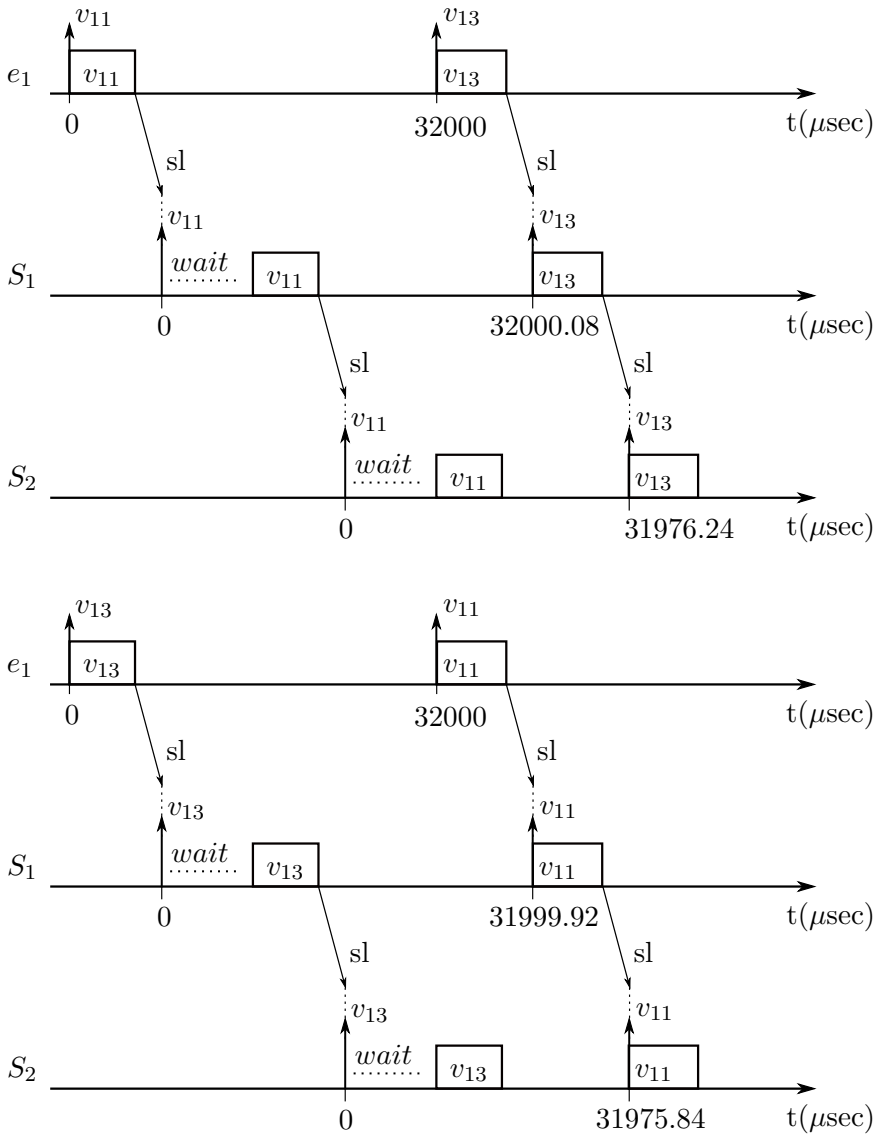


Figure 4.10 Relative offset between v_{11} and v_{13}

in the NC approach.

4.4.2 Integrating offset in NC arrival curve

The integration of offset in the NC approach was first proposed by Xiaoting Li [Li13] in the context of the switched Ethernet networks with FIFO scheduling. In this section, we extend this approach for switched Ethernet networks with DRR scheduling. We consider the definite offset (in μsec) for each flow at their respective end-node as given in Table

4.2.

Table 4.2 Definite offset for network (Figure 4.7)

v_i	$O^{e_1}_{d,i}$	v_i	$O^{e_2}_{d,i}$	v_i	$O^{e_3}_{d,i}$	v_i	$O^{e_4}_{d,i}$	v_i	$O^{e_5}_{d,i}$	v_i	$O^{e_6}_{d,i}$
v_{11}	0	v_2	32000	v_1	64000	v_3	32000	v_4	0	v_5	0
v_{13}	32000	v_{14}	0	v_9	32000	v_6	16000	v_7	32000	v_8	32000
				v_{10}	0	v_{12}	0				

The temporal separation of flows at end-node can be integrated into NC arrival curves. This integration is based on an aggregation technique.

The classical NC approach considers that all the flows are independent. Thus the arrival curves of all the flows crossing an output port are simply added in order to obtain the overall arrival curve at this port. It corresponds to the case where flows from each input arrive at the output port at the same time. This curve is illustrated in Figure 4.11 for v_{11} and v_{13} at e_1. The burst $(100 + 99) \times 8 = 1592$ bits at time $t = 0$ corresponds to arrivals of one frame from v_{11} and v_{13} at the same time. However, it has been shown

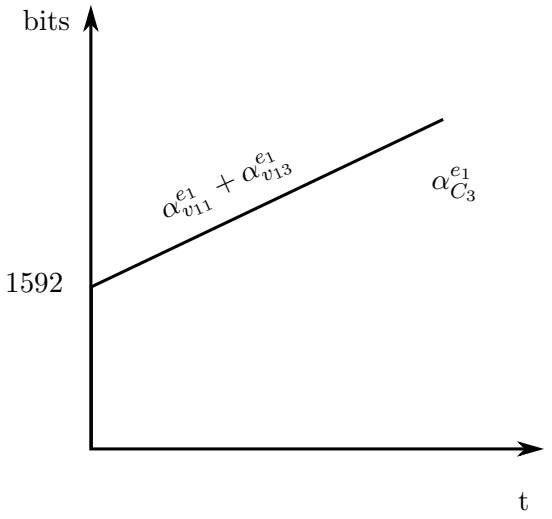


Figure 4.11 Overall arrival curve at e_1 in classical NC approach

earlier that there is a separation of at least 32000 μsec between these flows. Consequently, there will never be such a burst. Such separation can be taken into account by an aggregated curve of dependent flows (i.e. flows with minimum durations between them).

An *aggregated arrival curve* is represented by $\alpha^h_{v_b\{v_i\}}$, which is obtained when a flow v_b arrives before flow v_i at output port h, with temporal separation of $O^h_{r,b,i}$. In this case, $\alpha^h_{v_b\{v_i\}}$ is the sum of arrival curve $\alpha^h_{v_b}(\mathrm{t})$ and right-shifted arrival curve of v_i (i.e. $\alpha^h_{v_i}(t - O^h_{r,b,i})$).

At e_1, the aggregated arrival curve with benchmark flow $v_b = v_{11}$ is:

$$\begin{aligned}\alpha^{e_1}_{v_{11},\{v_{13}\}} &= \alpha^{e_1}_{v_{11}}(t) + \alpha^{e_1}_{v_{13}}(t - O^{e_1}_{r,11,13}) \\ &= \left(\frac{(99 \times 8)}{64000}t + (99 \times 8)\right) + \left(\frac{(100 \times 8)}{128000}(t - 32000) + (100 \times 8)\right) \\ &= (0.0123\ t + 792) + (0.00625\ (t - 32000) + 800)\end{aligned}$$

Similarly, the aggregated arrival curve with benchmark flow $v_b = v_{13}$ is

$$\alpha^{e_1}_{v_{13},\{v_{11}\}} = (0.00625\ t + 800) + (0.0123\ (t - 32000) + 792)$$

Both these aggregated arrival curves are shown (by dashed line curve) in Figure 4.12. The worst-case delay in each case is $D^{e_1}_{v_{11}} = 7.92\ \mu$sec and $D^{e_1}_{v_{13}} = 8\ \mu$sec as shown in Figure 4.12.

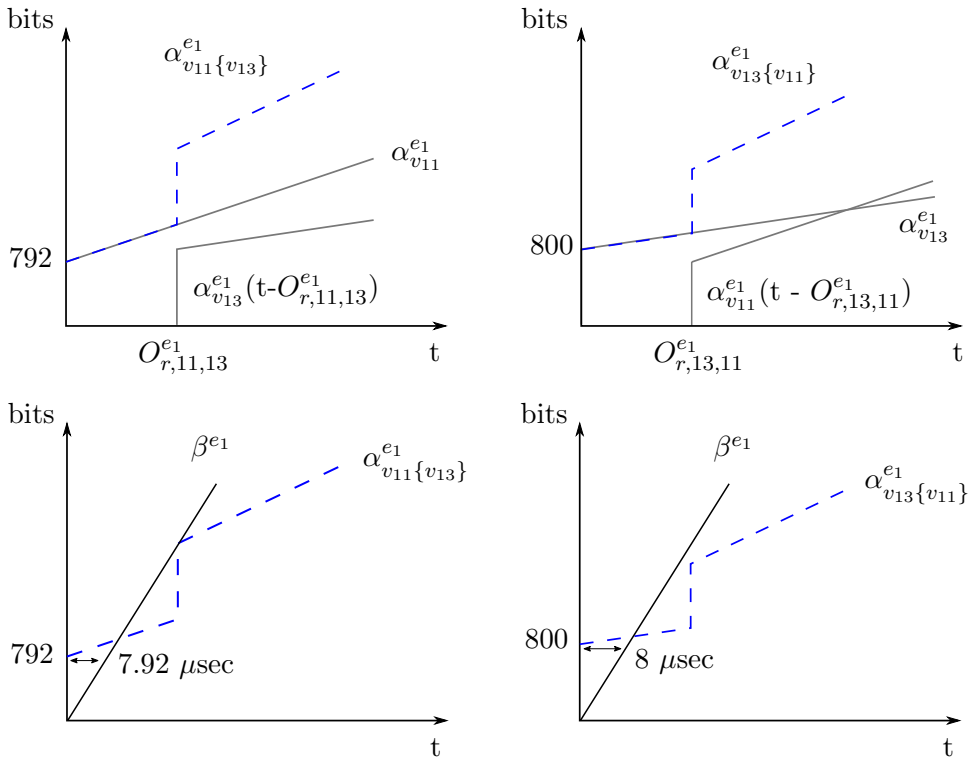


Figure 4.12 Aggregated arrival curves at e_1

At a switch output port, the aggregated arrival curves can be computed in similar manner. The overall arrival curve is obtained from the sum of the upper bound on aggregated traffic from each input. Thus the overall arrival curve of C_3 flows at S_1 is

computed as:

- Since, the flows in C_3 queue are arriving from 2 sources: v_{11} and v_{13} from e_1 and v_{14} from e_2, the aggregated traffic from each source is computed separately. Thus, we make two subsets $SS_1 = \{v_{11}, v_{13}\}$ and $SS_2 = \{v_{14}\}$.

- Compute the aggregated arrival curves in each subset SS_i and characterise the upper bound on traffic from this subset.

  For subset SS_1, we have,

$$\begin{aligned} O^{S_1}_{r,11,13} &= O^{e_1}_{r,11,13} - (D^{e_1,S_1}_{max,11} - D^{e_1,S_1}_{min,13}) \\ &= 32000 - (7.92 - 8) = 32000.08\mu sec \\ O^{S_1}_{r,13,11} &= O^{e_1}_{r,13,11} - (D^{e_1,S_1}_{max,13} - D^{e_1,S_1}_{min,11}) \\ &= 32000 - (8 - 7.92) = 31999.92\mu sec \\ \alpha^{S_1}_{v_{11}\{v_{13}\}}(t) &= \alpha^{S_1}_{v_{11}}(t) + \alpha^{S_1}_{v_{13}}(t - O^{S_1}_{r,11,13}) \\ \alpha^{S_1}_{v_{13}\{v_{11}\}}(t) &= \alpha^{S_1}_{v_{13}}(t) + \alpha^{S_1}_{v_{11}}(t - O^{S_1}_{r,13,11}) \end{aligned}$$

  where,

$$\begin{aligned} \alpha^{S_1}_{v_{11}}(t) &= (0.0123 \times t) + 792 \\ \alpha^{S_1}_{v_{13}}(t) &= (0.00625 \times t) + 800 \end{aligned}$$

  In order to compute the worst-case delay, the overall traffic from each input has to be upper bounded. Since there are two possible aggregated arrival curves for subset SS_1, the traffic represented by these two aggregated curves must be upper bounded. It is obtained by taking the piecewise maximum of the aggregated arrival curves, thus, we have

$$\alpha^{S_1}_{SS_1} = \max\{\alpha^{S_1}_{v_{11}\{v_{13}\}}, \alpha^{S_1}_{v_{13}\{v_{11}\}}\}$$

  And for subset SS_2, since there is only one flow (v_{14}), the traffic upper bound is same as the one given by arrival curve of flow v_{14}. Thus, $\alpha^{S_1}_{SS_2} = \alpha^{S_1}_{v_{14}} = (0.0125 \times t) + 800$.

  These arrival curve are shown in Figure 4.13.

- Finally, the overall arrival curve at S_1 is the sum of the arrival curve of each subset. Thus, we have,

$$\alpha^{S_1}_{C_3} = \alpha^{S_1}_{SS_1} + \alpha^{S_1}_{SS_2}$$

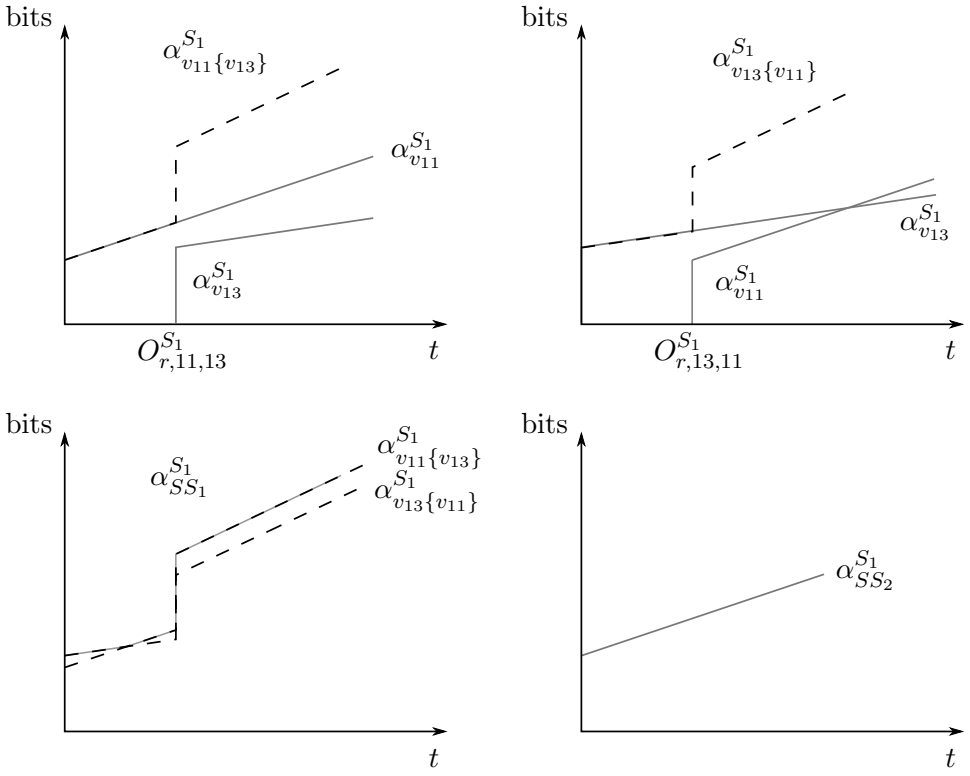


Figure 4.13 Aggregated arrival curves at S_1

as shown in Figure 4.14.

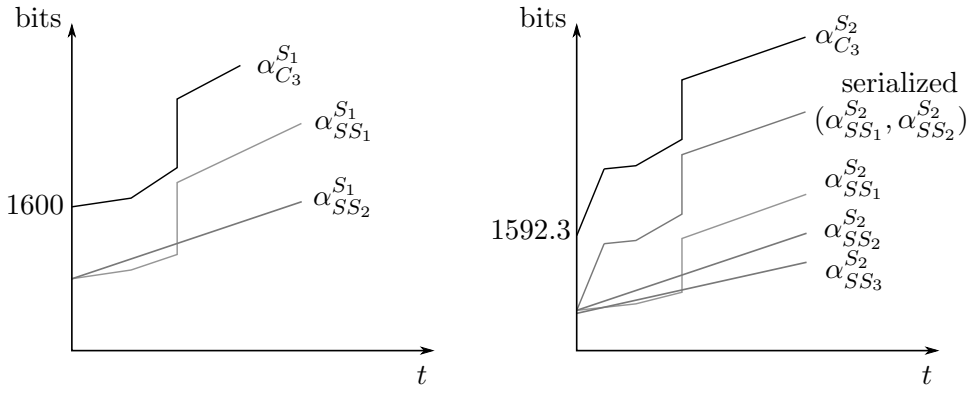


Figure 4.14 Overall arrival curve of C_3 flows at S_1 and S_2

This overall arrival curve can be used to compute the worst-case delay for C_3 flows

at S_1 as:

$$\begin{aligned} D_{C_3}^{S_1} &= \sup_{s\geq 0}(\inf\{\tau \geq 0 | \alpha_{C_3}^{S_1}(s) \leq \beta_{C_3}^{S_1}(s+\tau)\}) = 71.76\ \mu sec \\ where,\ \beta_{C_3}^{S_1} &= \rho_{C_3}^{S_1}[t - \Theta_{C_3}^{S_1} - sl]^+ = 50[t - 39.76]^+ \\ SL_{C_1}^{S_1}(D_{C_3}^{S_1}) &= 3976\ bits = 497\ bytes \\ L_{C_1}^{max,S_1}(D_{C_3}^{S_1}) &= \alpha_{C_1}^{S_1}(D_{C_3}^{S_1}) = 800\ bits = 100\ bytes \\ D_{C_3,opt}^{S_1} &= D_{C_3}^{S_1} - \frac{SL_{C_1}^{S_1}(D_{C_3}^{S_1}) - L_{C_1}^{max,S_1}(D_{C_3}^{S_1})}{R} \\ &= 71.76 - 31.76 = 40\ \mu sec \end{aligned}$$

The overall arrival curve at S_2 can be computed similarly. We get 3 subsets: $SS_1 = \{v_{11}, v_{13}\}$, $SS_2 = \{v_{14}\}$ and $SS_3 = \{v_{12}\}$. Since the flows from subset SS_1 and SS_2 are arriving at S_2 from same input (S_1), they are serialised as

$$\min\{((R \times t) + \max\{b_{v_{11}}^{S_2}, b_{v_{13}}^{S_2}, b_{v_{14}}^{S_2}\}), (\alpha_{SS_1}^{S_2} + \alpha_{SS_2}^{S_2})\}$$

Therefore, the overall arrival curve of C_3 flows at S_2 is

$$\begin{aligned} \alpha_{C_3}^{S_2} &= \min\{((R \times t) + \max\{b_{v_{11}}^{S_2}, b_{v_{13}}^{S_2}, b_{v_{14}}^{S_2}\}), (\alpha_{SS_1}^{S_2} + \alpha_{SS_2}^{S_2})\} + \alpha_{SS_3}^{S_2} \\ where,\ \alpha_{SS_1}^{S_2} &= \max\{\alpha_{v_{11}\{v_{13}\}}^{S_2}, \alpha_{v_{13}\{v_{11}\}}^{S_2}\} \\ \alpha_{SS_2}^{S_2} &= \alpha_{v_{14}}^{S_2} = (0.125 \times t) + 800.3 \\ \alpha_{SS_3}^{S_2} &= \alpha_{v_{12}}^{S_2} = (0.0123 \times t) + 792 \\ \alpha_{v_{11}}^{S_2}(t) &= (0.0123 \times t) + 792.3 \\ \alpha_{v_{13}}^{S_2}(t) &= (0.00625 \times t) + 800.15 \end{aligned}$$

These arrival curves are also shown in Figure 4.14. The worst-case delay for C_3 flows at S_2 is 95.77 μsec.

4.5 Evaluation of optimised NC approach

In this section, we compare the delay bounds computed by classical NC approach with those computed by optimised NC approach. For this purpose, we consider the network example shown in Figure 4.7.

The end-to-end delay computed for the given network using classical NC approach and optimised NC approach are shown in Table 4.3. First, the delays are computed without

Table 4.3 End-to-end delays comparison

Flows (v_i)	End-to-end delay (μsec)		
	classical NC	optimised NC	
		without offset	with offset
v_1	214.99	143.62	111.66
v_2	262.83	175.63	143.68
v_3	214.91	143.54	111.66
v_4	206.99	135.62	111.74
v_5	206.99	135.62	111.74
v_6	206.9	135.58	127.56
v_7	198.98	127.66	127.48
v_8	198.98	127.66	127.48
v_9	206.98	135.66	127.56
v_{10}	206.98	135.66	127.56
v_{11}	246.77	167.55	143.37
v_{12}	175.01	143.46	103.69
v_{13}	246.77	167.55	143.77
v_{14}	246.85	167.63	143.77

considering any offset at end-nodes. In this case, the delays computed using optimised NC approach are, on average, 32.7 % lower than those computed by the classical NC approach. On adding offsets in the computation the results in optimised NC approach are further improved by 13.3 %.

4.6 Conclusion

In this chapter, we have proposed an optimised NC approach which mitigates the pessimism involved in classical NC approach. In the next chapter, we perform an industrial case study which illustrates the significant reduction in pessimism by our optimised NC approach.

Chapter 5

Case Study on An Industrial Real-Time Switched Ethernet Network : AFDX Network

Contents

5.1 Introduction 103
5.2 Context of AFDX network 104
5.2.1 AFDX network and flow model 104
5.2.2 Reference AFDX network 105
5.2.3 Flow differentiation 107
5.3 WCTT analysis based on classical NC approach 108
5.3.1 First-In-First-Out (FIFO) scheduling 108
5.3.2 Static Priority Queuing (SPQ) scheduling 111
5.3.3 Deficit Round Robin (DRR) scheduling 111
5.4 WCTT analysis based on optimised NC approach 114
5.5 Conclusion 116

5.1 Introduction

In previous chapters, an optimal credit assignment algorithm based on classical NC approach has been proposed to achieve efficient bandwidth sharing in DRR based real-time switched Ethernet networks and an optimised Network Calculus approach has been

proposed to upper bound the end-to-end delays in such networks. The evaluation of these approaches on an industrial application is important in order to show the improvement provided by these approaches.

In this chapter, we perform a case study on an industrial real-time switched Ethernet network: AFDX network. We first show the worst-case delay analysis on existing AFDX network architecture (based on FIFO scheduling) in presence of flows of mixed-criticality and then we compare the SPQ and DRR scheduler based AFDX network architecture under the similar flow constraints while using our optimal credit allocation algorithm. At last, we show the improvements in delay computation using our optimised NC approach.

In Section 5.2 the AFDX network and its industrial configuration is presented. In Section 5.3 and 5.4 a WCTT analysis of the given AFDX network is performed using classical NC approach and optimised NC approach respectively. Section 5.5 concludes the chapter.

5.2 Context of AFDX network

Avionics Full-Duplex (AFDX) is a typical real-time switched Ethernet network. AFDX network was previously described in section 1.3.3. AFDX was developed by Airbus for the modern aircraft system to address real-time issues in safety-critical avionics developments. It was first implemented in Airbus A380 aircraft. The main goals of the AFDX network were to provide reliable (guaranteed delivery) and deterministic (bounded delays) communication. AFDX standard is defined in Aeronautical Radio Incorporated (ARINC) specification number 664 part-7. The network and flow model of this network and its industrial configuration are given in the following paragraphs.

5.2.1 AFDX network and flow model

An AFDX network consists of end-systems (source and destination of flows) interconnected by switches via full-duplex Ethernet links. There exist no synchronisation clock between end-systems. To provide a deterministic communication behaviour, each end-system send compliant traffic (shaping) and the switch directly lined to end-system enforces compliance (policing).

Each flow transmitted on this network is called a Virtual Link (VL). A VL is standardized by ARINC 664 as a virtual communication channel, which is the basis of multicast AFDX flows. Each VL is characterized by a Bandwidth Allocation Gap (BAG), which is a minimum time interval between consecutive frames of VL. The BAG value ranges in powers of 2 from 1 ms to 128 ms. The frame lengths (l_i^{max} and l_i^{min}) in a VL

(v_i) are constrained by the Ethernet standard. Thus, an end-system can transmit at most l_i^{max} bytes on v_i in an interval of BAG_i. Therefore the maximum long-term traffic arrival rate for v_i in bits per second, denoted by r_i, is:

$$r_i = \frac{l_i^{max}}{BAG_i}$$

The maximum load induced by v_i on a switch output port is upper bounded by the maximum arrival rate r_i and the link rate R, as :

$$\frac{l_i^{max}}{R \times BAG_i}$$

Existing AFDX network employs switches with $R = 100$ Mbps. Each output port is controlled by a Strict Priority/First-In First-Out (SP/FIFO) scheduler with up to 2 levels of priority (high and low).

Since there is no global clock, there is no synchronization between the end systems. Thus, each end-system schedules the VLs independently. However, the periodic flows generated at the same end-system are scheduled based on a local clock and therefore they are called locally synchronized and they are dependent. This scheduling of flows at end-system introduces a temporal separation between flows and hence reduce the instantaneous traffic in the network.

5.2.2 Reference AFDX network

Figure 5.1 shows an architecture of an industrial avionics configuration based on the network architecture of A380 aircraft. It includes 96 end-systems interconnected by two redundant networks. These redundant networks are exact replicas of each other with separate power sources and different routing of cables. This redundancy provides better communication guarantees. Each end-system emits each frame on the two networks simultaneously. The destination end-system uses First-Valid-Win mechanism i.e. on arrival of a frame the receiver considers only the first of the two packets issued in parallel.

Each network has 8 switches forwarding 984 VLs on a total of 6276 paths (multicast). Table 5.1 shows the dispatching of VLs among BAGs and maximum frame lengths. More than 80% of flows have small frame length (up to 600 bytes).

In an avionic application, the path of a VL is limited to 4 switches. The goal of this limitation is to minimize the overall delay experienced by VL. Table 5.2 shows the distribution of paths according to their lengths, expressed in terms of the number of traversed switches.

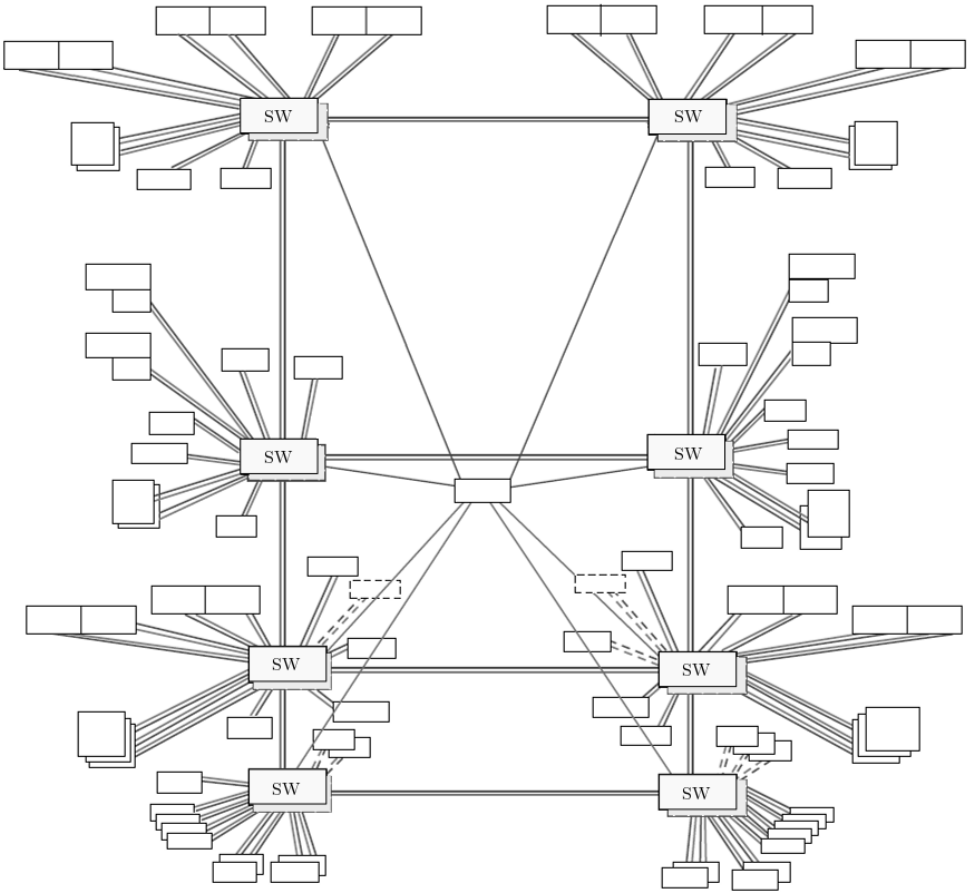


Figure 5.1 AFDX network architecture

Table 5.1 Flow characteristics (a) BAG (b) Frame length

BAG (ms)	No. of VLs
2	20
4	40
8	78
16	142
32	229
64	220
128	255

(a)

Frame Length (bytes)	No. of VLs
84-200	278
201-400	396
401-600	157
601-900	69
901-1200	28
1201-1500	51
> 1500	5

(b)

Table 5.2 Length of VL paths

Path length (No. of switches)	No. of paths
1	1780
2	2807
3	1436
4	253

5.2.3 Flow differentiation

We arbitrarily distribute the 984 flows of the given configuration among 2 critical classes: a class C_1 for critical flows with deadline 12573 μsec and a class C_2 for less-critical flows with deadline 50292 μsec. We consider flows with small BAG (up to 16 msec) in critical class C_1 since these flows can have smaller delay constraints (in this case 12573 μsec) as compared to the flows with larger BAG. The remaining flows are considered in less-critical class C_2. The characteristics of these classes are also summarised in Table 5.3.

Table 5.3 VL parameters in industrial configuration

Flow count	*BAG* Range (*msec*)	Frame length range (*bytes*)	Class	Deadline (μsec)
280 (1681 paths)	2 – 16	84 – 1497	Critical C_1	12573
704 (4595 paths)	32 – 128	84 – 1535	Less-Critical C_2	50292
Additional flows				
10 – 70 (501 – 699 paths)	4 – 8	84 – 1355	Best-Effort C_3	-

We arbitrarily introduced some additional flows which share the path with each critical flow on at least one switch output port. The idea is to interfere with the critical flows and evaluate the impact on worst-case end-to-end delay in the network under different scheduling policies. The added flows are considered as non-critical (best-effort) flows characterised by class C_3 shown in Table 5.3. In the next section, different scenarios are presented in the WCTT analysis of the given network configuration.

5.3 WCTT analysis based on classical NC approach

In this section, we perform the WCTT analysis of the given network using classical NC approach under FIFO, SPQ and DRR scheduling. The main objective of this section is to, first, show the effect of additional non-critical flows on existing critical flows in terms of change in end-to-end delays, second, to compare the service provided to additional non-critical flows in case of each scheduling policy.

5.3.1 First-In-First-Out (FIFO) scheduling

Initially, the given network is assumed to use FIFO scheduling in each switch output port. In the following paragraphs, first, we compute the worst-case delay in the original network configuration, i.e. without C_3 flows. Then in the second step, we consider different scenarios to compute the worst-case delay in the presence of C_3 flows. In each scenario, the number of C_3 flows is increased to analyse its impact on C_1 and C_2 flows.

Without additional flows: The worst-case end-to-end delays computed on all the paths (1681 paths of C_1 flows and 4595 paths of C_2 flows) for each flow in the network are plotted in Figure 5.2, where each unit on the x-axis represents a flow path and y-axis represents the worst-case end-to-end delay corresponding to this path. For illustration purpose in Figure 5.2, the paths are sorted in increasing order of delay. The plot can be read as: there are at least 500 flow paths in C_1 where the delay is less than 3000 μsec, and there are at least 1000 flow paths in C_1 whose delay is between 2000 μsec and 8000 μsec. The maximum delay in the network is 12572.6 μ sec. It can be seen in Figure 5.2 that the delay in each flow path is less than the flow deadline.

A more detailed observation of number of flow paths in certain range of end-to-end delays is given in Table 5.5 by column "No. of C_3 flows = 0" to compare it with other cases discussed later in this section.

With additional flows: Now, let us consider the additional non-critical flows (best-effort) in the given network (the characteristics of these flows are given in Table 5.4). To analyse the impact of these additional flows on the delays of C_1 and C_2 flows, we consider 7 different scenarios (Table 5.4). In the first scenario, we consider only 10 flows of class C_3 and compute the delays on all flow paths. In the second scenario, we consider 20 flows of class C_3 (10 flows from previous scenario + 10 new flows) and compute the delays on all flow paths. Similarly, in the remaining scenarios, we continue to add 10 more flows of class C_3 and compute the delays.

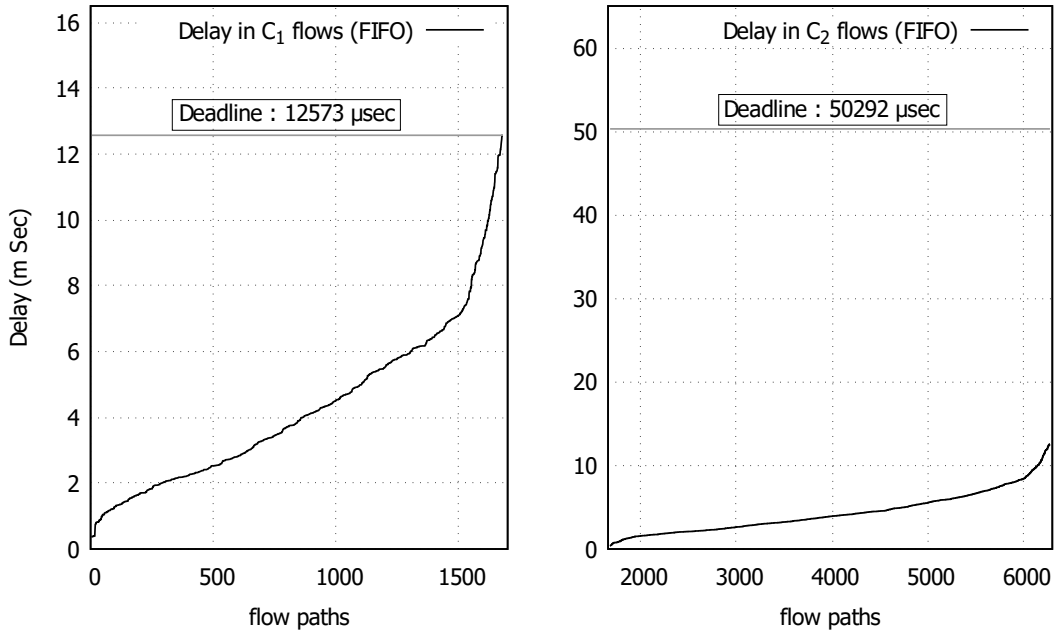


Figure 5.2 End-to-end delay bounds in C_1 and C_2 flows under FIFO scheduling

Table 5.4 VL parameters in best-effort class C_3

Case	1	2	3	4	5	6	7
Flow count	10	20	30	40	50	60	70
Path count	501	568	584	616	640	663	699
BAG Range ($msec$)	4–8						
Frame length range ($bytes$)	84-934	84–1355					

The results obtained in each scenario are compared in Table 5.5.

With the increase in the number of C_3 flows, not surprisingly, the delay is increased for critical flows (C_1 and C_2) as C_3 flows share the same FIFO waiting queues with critical flows and eventually increase the waiting delay in the queue. For instance, in $\frac{287+576}{1681} = 51.33\%$ paths of C_1 flows the delay is less than 4000 μsec and this percentage is reduced to $\frac{192+581}{1681} = 45.98\%$ on of introducing 70 flows in C_3 flows.Moreover, there are a significant number of flows of critical class C_1 that exceeded their deadlines. In presence of 70 C_3 flows, there are as many as 43 paths on which C_1 flows missed their deadlines. A similar impact is present on C_2 flows however these flows remain within their delay constraints as their deadline is quite high.

Table 5.5 End-to-end delay bounds in C_1 and C_2 flows under FIFO scheduling in presence of additional non-critical (C_3) class flows

	No. of C_3 flows							
	0	10	20	30	40	50	60	70
Delay range (μsec)	No. of C_1 flow paths							
<= 2000	287	268	256	248	224	221	202	192
>2000 <=4000	576	567	552	558	570	571	581	581
>4000 <=6000	441	437	402	393	391	376	366	360
>6000 <=8000	249	277	324	326	322	330	340	340
>8000 <=10000	69	70	70	79	95	102	99	111
>10000 <=12573	59	58	68	59	59	57	62	54
>12573 (missed deadline)	**0**	**4**	**9**	**18**	**20**	**24**	**31**	**43**

	No. of C_3 flows							
	0	10	20	30	40	50	60	70
Delay range (μsec)	No. of C_2 flow paths							
<= 2000	691	650	620	610	590	578	562	546
>2000 <=4000	1690	1639	1644	1650	1649	1648	1647	1640
>4000 <=6000	1185	1201	1122	1098	1091	1081	1069	1071
>6000 <=8000	653	674	713	721	695	707	717	705
>8000 <=10000	247	287	327	343	383	381	382	399
>10000 <=12000	129	113	117	113	123	132	132	125
>12000 <=50292	0	31	52	60	64	68	86	109
>50292 (missed deadline)	0	0	0	0	0	0	0	0

5.3.2 Static Priority Queuing (SPQ) scheduling

Let us now perform the WCTT analysis on the given network while considering SPQ scheduling at each switch output port. An SPQ scheduler differentiates flow traffic based on a predefined priority. We assign a priority P_{C_i} to each flow class based on the decreasing order of criticality: $P_{C_1} > P_{C_2} > P_{C_3}$. Let us analyse the end-to-end delays on C_1 and C_2 flows by considering the 7 scenarios, as discussed earlier and shown in Table 5.4. The results obtained in each scenario are compared in Table 5.6.

As shown in Table 5.6, there is not much variation in delays of critical classes C_1 and C_2 on increasing the number of additional C_3 flows, this is due to the fact that an SPQ scheduler manages separate queues dedicated to each priority level and they are served in descending order of priority, thus, a low priority C_3 flow cannot start transmission if there is a pending high priority (C_1 and C_2) flow. However, there is a small impact of low priority flow on high priority flow due to non-preemption delay. The non-preemption delay is limited to transmission time of the largest frame of low priority. In the given network configuration, the maximum non-preemption delay can be $\frac{934\times8}{100} = 74.72$ μsec (case 1) or $\frac{1355\times8}{100} = 108.4$ μsec (cases 2 to 7). In certain paths, this non-preemption delay leads to a significant increase in delay of C_1 flows such that these flows miss their deadline (see Table 5.6).

Let us now compare the worst-case end-to-end delays of C_3 flows in SPQ and FIFO scheduling. Since the number of C_3 flows is increased in each scenario, we only compare the average and maximum delay in case of both schedulers. The comparison result is shown in Figure 5.3. As illustrated in Figure 5.3 the delays in C_3 flows are higher in case of SPQ as compared to FIFO scheduling. Indeed, in order to lower the impact of low priority flows on high priority flows, the SPQ scheduler degrades the quality of service for low priority flows. Thus, it is unfavourable to non-critical C_3 flows.

5.3.3 Deficit Round Robin (DRR) scheduling

Now, we consider the given network with DRR scheduling at each switch output port. As we have seen in Chapter 3, in DRR scheduling, the delay in each class flows is affected by the distribution of quantum among these classes. We have also proposed an algorithm (Algorithm 2) in Section 3.4.2 to optimise the quantum distribution such that it leads to minimum delay in non-critical flows while allowing the critical flows to respect their deadlines.

Let us evaluate this algorithm on the given industrial configuration. For that purpose, we compute the optimised quantum distribution using our algorithm, which leads to the

Table 5.6 End-to-end delay bounds in C_1 and C_2 flows under SPQ scheduling in presence of additional non-critical (C_3) class flows

	No. of C_3 flows						
	10	20	30	40	50	60	70
Delay range (μsec)	No. of C_1 flow paths						
<= 2000	277	271	271	266	264	256	256
>2000 <=4000	579	574	574	579	581	589	587
>4000 <=6000	438	428	418	415	415	415	414
>6000 <=8000	257	271	281	284	284	284	286
>8000 <=10000	71	70	68	68	68	68	68
>10000 <=12573	55	63	64	64	64	64	65
>12573 (missed deadline)	**4**	**4**	**5**	**5**	**5**	**5**	**5**

	No. of C_3 flows						
	10	20	30	40	50	60	70
Delay range (μsec)	No. of C_2 flow paths						
<= 2000	674	658	658	658	658	648	648
>2000 <=4000	1680	1685	1679	1673	1673	1683	1676
>4000 <=6000	1166	1125	1124	1128	1123	1123	1127
>6000 <=8000	665	696	702	704	707	707	705
>8000 <=10000	274	287	288	288	290	290	293
>10000 <=12000	110	103	95	95	95	95	97
>12000 <=50292	26	41	49	49	49	49	49
>50292 (missed deadline)	0	0	0	0	0	0	0

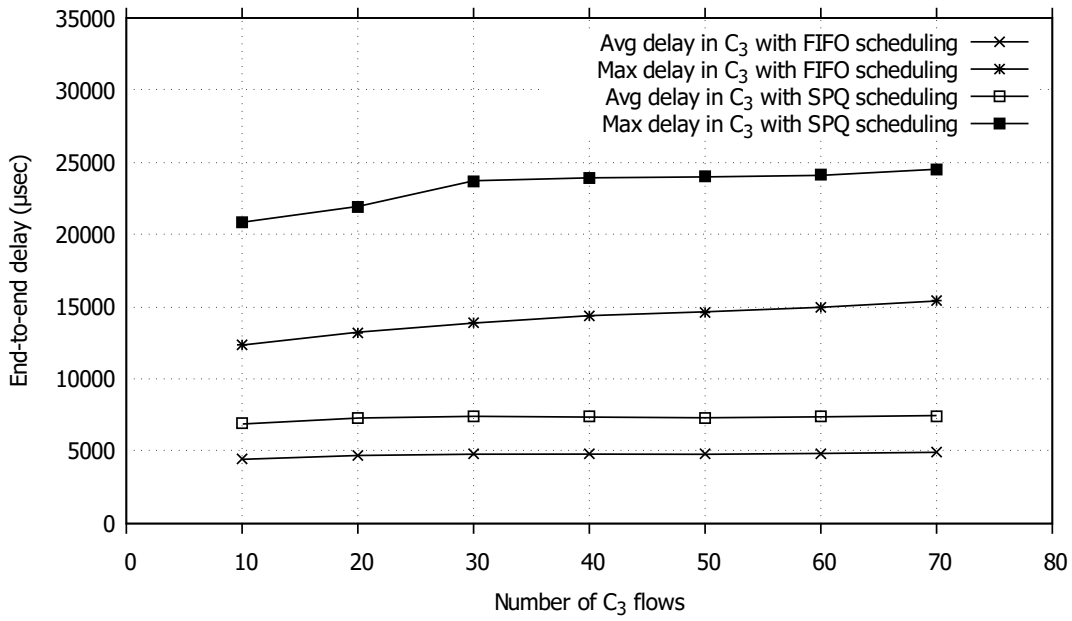


Figure 5.3 End-to-end delay comparison in C_3 flows under FIFO and SPQ

distribution of a sum of quanta $Q = 6340$ bytes as

$$Q_{C_1} = 3659\ bytes,\ Q_{C_2} = 1535\ bytes\ and\ Q_{C_3} = 1146\ bytes$$

in this case, the bandwidth allocated to each class is

$$\rho_{C_1} = 57.71\ bits/\mu sec,\ \rho_{C_2} = 24.21\ bits/\mu sec,\ \rho_{C_3} = 18.07\ bits/\mu sec$$

On computing the end-to-end delays with this quantum distribution, we get maximum delays in each class as $D_{C_1}^{max} = 12493.8$ μsec, $D_{C_2}^{max} = 50197.7$ μsec, and $D_{C_3}^{max} = 5815.58$ μsec. These results correspond to the case where 10 C_3 flows are considered. Similarly, we apply our algorithm to compute delays in all the 7 scenarios discussed earlier (Table 5.4).

Since the computation of the quantum distribution in Algorithm 2 aims to provide sufficient quantum to critical classes so that their deadlines are respected and since the traffic in critical classes is not changed in the given cases, we get the same distribution for all the cases. So, the residual bandwidth for C_3 flows in each case is 18.07 bits/μsec.

The worst-case end-to-end delays of C_3 flows computed with DRR scheduling are compared with SPQ and FIFO scheduling in Figure 5.4.

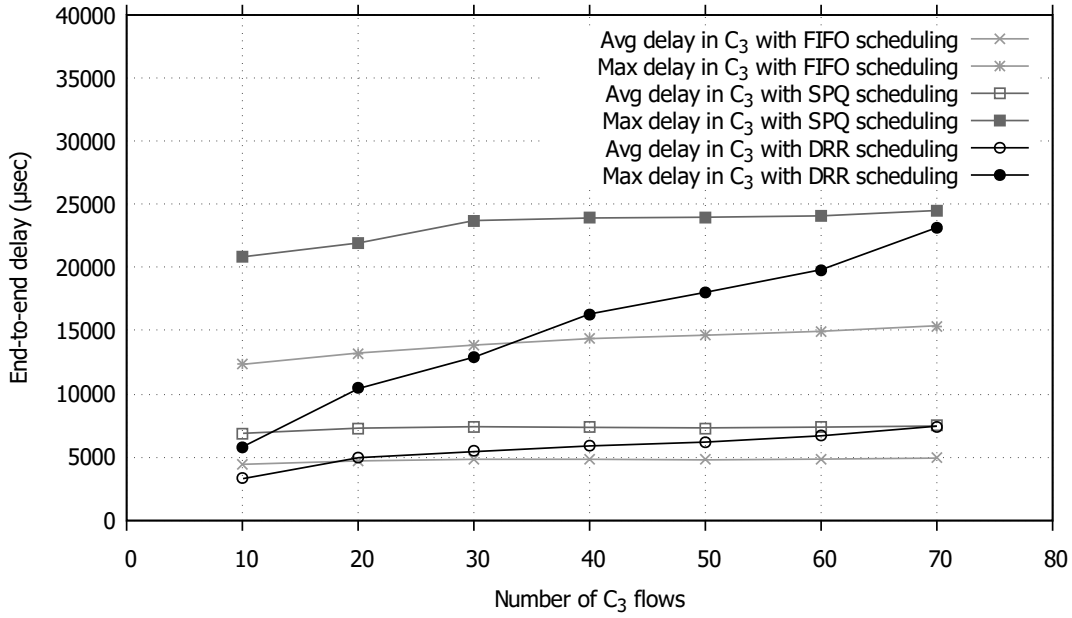


Figure 5.4 End-to-end delay comparison in C_3 flows under FIFO, SPQ and DRR

As illustrated in Figure 5.4 the delays in C_3 flows in DRR scheduling are much lower than that in SPQ and FIFO scheduling for the lower values of C_3 flows. With the increase in the number of C_3 flows the delays observed with DRR scheduling are increased, which is obvious since the residual bandwidth for C_3 flows is same in each case. However, in all the scenarios DRR scheduling allows to respect the delay constraints in critical classes (C_1 and C_2), which is not the case in SPQ and FIFO scheduling.

Therefor, it can be concluded that DRR scheduler outperforms the SPQ and FIFO scheduling when appropriate quantum is assigned to different classes. For the given network configuration, our algorithm is able to successfully compute such a quantum assignment.

5.4 WCTT analysis based on optimised NC approach

We have shown in Chapter 4 that the classical NC approach for DRR scheduler can be significantly pessimistic, depending on the traffic in different classes. We have also proposed an optimised NC approach in order to reduce this pessimism. In the following paragraphs, we evaluate this reduction in pessimism for the given industrial configuration.

Figure 5.5 shows the comparison of delays computed using classical NC and optimised

NC approach, where each unit on the x-axis represents a flow path and y-axis represents the worst-case end-to-end delay corresponding to this path. To compare the delays in each path $\mathcal{P}_x$ of each flow v_i, the delay upper bound $D_{i,NC}^{\mathcal{P}_x}$ computed by the classical NC approach is taken as the reference value and it is normalized to 100. Then the delay upper bound $D_{i,NC_{opt}}^{\mathcal{P}_x}$ computed by optimized NC approach is normalized as

$$D_{i,NC_{opt},norm}^{\mathcal{P}_x} = \frac{D_{i,NC_{opt}}^{\mathcal{P}_x}}{D_{i,NC}^{\mathcal{P}_x}} \times 100$$

The comparison in Figure 5.5 corresponds to the scenario where only 10 flows are considered in C_3 and the quanta $Q_{C_1} = 3659$ bytes, $Q_{C_2} = 1535$ bytes and $Q_{C_3} = 1146$ bytes (obtained from Algorithm 2) are assigned to C_1, C_2 and C_3 respectively. For illustration purpose, the paths are sorted in increasing order of $D_{i,NC_{opt},norm}^{\mathcal{P}_x}$ in each class. In critical classes C_1 and C_2, the average gain is 15.37% and 53.5% respectively,

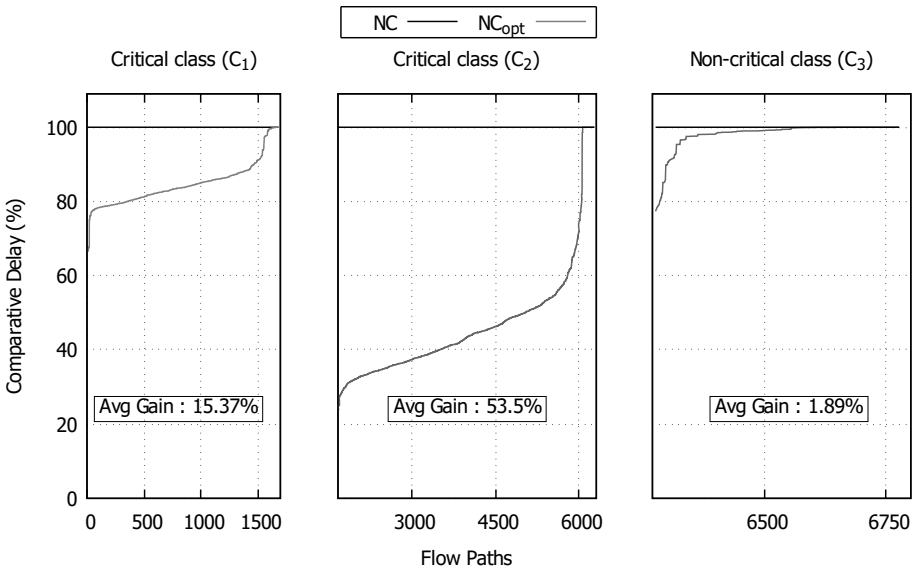


Figure 5.5 Comparison of end-to-end delay bounds computed by classical NC and optimised NC approach

which means that on most of the paths the load in switch output ports is not equally shared between classes and thus the interference of competing classes is much less than what is considered by classical NC approach. Since our optimised approach takes into account the effective traffic in each class the computed delays are much tighter. For C_3 the gain is relatively smaller, which is intuitive since the traffic in C_3 is quite less and a flow in C_3 can be delayed in each DRR round during its active period, depending on the

effective traffic in competing classes.

On the total number of paths in the given configuration, the average gain observed in the delay bound computed by our optimised NC approach is around 40.24%. This is a significant improvement which shows that the proposed optimisations are relevant on an industrial configuration.

Effect of pessimism on the results of Algorithm 2: As shown in Section 3.4.2, Algorithm 2 computes the smallest quantum share Q_{C_x} for each critical class C_x that assures the computed delay bound (based on classical NC appraoch) is near (and smaller) to its deadline. As we have seen in Figure 5.5 that the delay bound computed by classical NC approach can be significantly pessimistic, in that case, Q_{C_x} may not be the smallest quantum share for C_x. It means that there remains a possibility to decrease the quantum share for C_x (eventually increasing the residual quantum share for non-critical class) while still respecting its deadline. This can be achieved with the help of optimised NC approach.

However, the optimised NC approach cannot be directly integrated in Algorithm 2 as it is based on the assumption that the WCTT analysis is carried out independently for each class, i.e. in order to compute the end-to-end delay bound in a class C_x, only Q_{C_x} (a portion of Q) and Q (the sum of quantum distributed among all the classes) are needed. This assumption does not hold for the optimised NC approach. In order to compute tight delay bounds, the optimised NC approach takes into account the traffic from competing classes, which is affected by the distribution of Q in these classes.

We aim to resolve this problem in our future works by extending Algorithm 2 such that its quantum distribution output is improved by using optimised NC approach.

5.5 Conclusion

In this chapter, we have presented the AFDX network and we performed an evaluation on this network to illustrate the results of the credit optimisation algorithm and the optimised NC approach proposed in previous chapters.

The results show that the credit optimisation algorithm successfully gives the quantum distribution that allows the guarantee on critical flows while minimising the delays for non-critical flows. We have also shown that the service provided by DRR scheduler is better than that by the SPQ scheduler in a mixed-criticality flow environment. We have also illustrated that the pessimism in the delay upper bound computation is significantly reduced, up to 40%, by our optimised NC approach.

In the next chapter, we present the WCTT analysis of a real-time switched Ethernet with WRR scheduling and compare it with the DRR scheduler based network.

Chapter 6

WCTT Analysis in a Switched Ethernet Network with WRR Scheduler

Contents

6.1 Introduction 119
6.2 Overview of WRR schedulers 120
6.2.1 WRR Algorithm 120
6.2.2 Bandwidth sharing in WRR scheduler 123
6.3 Worst-Case Traversal Time analysis 127
6.3.1 WRR scheduler latency 127
6.3.2 Network Calculus approach 128
6.4 Case study on industrial AFDX network 135
6.4.1 WCTT analysis on an industrial size AFDX network configuration 135
6.4.2 WCTT analysis on a modified configuration of an industrial size AFDX network 139
6.5 Conclusion 147

6.1 Introduction

In this thesis, until now we have focused on DRR scheduling based switched Ethernet network. The DRR scheduling has the main advantage of high fairness in terms of

bandwidth utilisation, thanks to its ability of take into account the deficit of service per class per round so that the average service received by a class remains proportional to its predefined quantum. But, this fairness comes at a cost of high implementation complexity and elevated scheduler latency. The Weighted Round Robin (WRR) scheduling is another round robin discipline which offers efficient service at low implementation complexity and a low scheduler latency. However, as we will see in this chapter, WRR is less fair as compared to DRR in the case where the flow classes have a wide range of frame sizes since it is limited by its inability to take frame size into consideration. But, it can still provide fair service to flow classes that have more or less homogeneous characteristics, which is a not so uncommon scenario in industrial settings. For this reason, we analyse the WRR scheduling based network architecture for real-time industrial application.

In this chapter, we extend our optimised NC approach for WCTT analysis of WRR based real-time switched Ethernet network. We also perform a case study on industrial AFDX network configuration considering WRR scheduler at its switch output ports. In the end, a comparison of the delay bounds in WRR scheduling based network to those in DRR scheduling based network is also presented.

In this chapter, we first present the overview of WRR scheduling in Section 6.2. In Section 6.3 we show the WCTT analysis based on optimised NC approach. In Section 6.4 we perform a case study on industrial size AFDX network to compare the worst-case delays under WRR and DRR scheduling followed by a conclusion in Section 6.5.

6.2 Overview of WRR schedulers

We assume a real-time switched Ethernet network architecture similar to the one considered in previous chapters except that, in this chapter, we consider WRR scheduling at each switch output port. The working principle of WRR scheduling is described in the following paragraphs.

6.2.1 WRR Algorithm

The basic idea of a WRR scheduler is to assign a weight $W^h_{C_x}$ to each active class C_x at a given switch output port h. $W^h_{C_x}$ is the number of frames and it corresponds to the maximum credit that can be assigned to C_x in each scheduling round at h. So, in any round, at max $W^h_{C_x}$ frames can be served from C_x queue.

Algorithm 3, shows the basic principle of WRR scheduler serving n traffic classes. The scheduler selects class queues in a round robin order (line 2-10). The selected class queue is represented by counter i. In each round, the inactive classes are ignored (line 3)

and each active class receives a credit of W_i^h (line 4). From the selected class queue, the frames are sent as long as the queue is not empty and there is some remaining credit (line 5-9). If the selected class queue becomes empty before W_i^h frames are transmitted, remaining transmission opportunities (i.e. credit) are lost.

Algorithm 3: WRR Algorithm

```
   Input : Per class weight: W_1^h ... W_n^h (Integer)
   Data: Per class credit: credit_1^h ... credit_n^h (Integer)
   Data: Counter: i (Integer)
 1 while true do
 2     for i = 1 to n do
 3         if isActive(C_i) then
 4             credit_i^h ← W_i^h;
 5             while (isActive (C_i)) and (credit_i^h > 0) do
 6                 send(headFrame(C_i));
 7                 credit_i^h ← credit_i^h − 1;
 8                 remove(headFrame(C_i));
 9             end
10     end
11 end
```

Let us illustrate the WRR service with an example of a switch output port serving 20 flows from 3 classes at link rate 100 bits/μsec, shown in Figure 6.1. In this example, the flows v_1 to v_4 are characterised as C_1, v_5 to v_{12} as C_2 and v_{13} to v_{20} as C_3. The temporal and size constraints on each flow is given in Table 6.1.

Table 6.1 Flow specifications

Flows	T_i (msec)	Frame lengths (bytes)
v_5, v_6, v_{12}, v_{20}	128	80–100
$v_1, v_7, v_8, v_9, v_{17}$	128	80–99
$v_2, v_4, v_{10}, v_{13}, v_{14}, v_{15}$	64	80–100
$v_3, v_{11}, v_{16}, v_{18}, v_{19}$	64	80–99

One possible scenario for WRR scheduling at the given switch output port is shown in Figure 6.2 where classes C_2 and C_3 are assigned a credit of $W_{C_2} = W_{C_3} = 2$ frames per round and C_1 is assigned $W_{C_1} = 1$ frame per round.

Initially, all the class queues are empty. At t_1, 6 frames arrive: from v_5 and v_{12} in C_2 and v_4, v_3, v_2 and v_1 in C_1 (in the given order). The decision about the class that

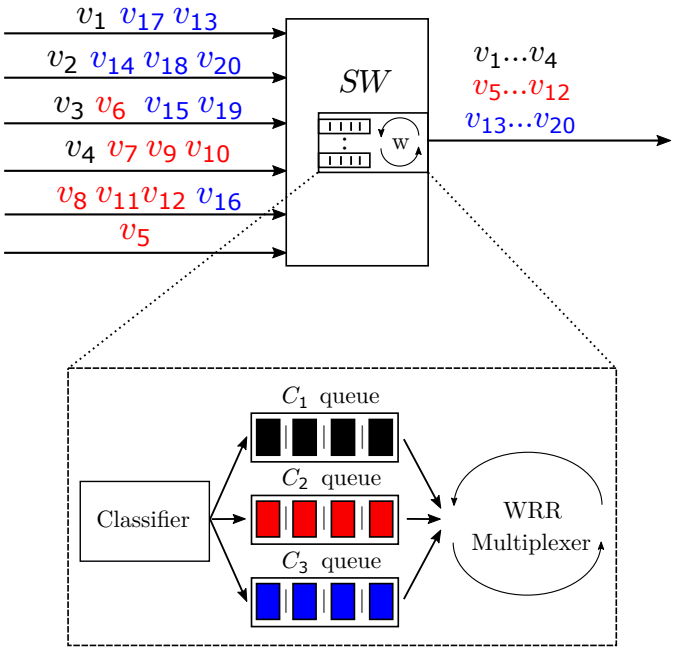


Figure 6.1 An example switch with WRR scheduling at output port

will be served first depends upon the arrival instant of the frames in that class buffer. Let us assume C_2 is selected first and it receives a credit $W_{C_2} = 2$ frames. C_2 consumes its full credit in the transmission of v_5 and v_{12} frames. Meanwhile 9 more frames have arrived: from v_6, v_7, v_8 and v_9 in C_2 and from v_{14}, v_{15}, v_{16}, v_{17} and v_{18} in C_3. Since C_2 has no credit left, its newly arrived frames cannot be served in this round. The scheduler moves to the next active class C_3. C_3 receives a credit $W_{C_3} = 2$ frames and consumes it in transmission of v_{14} and v_{15} frames. The next active class is C_1 which gets a credit of $W_{C_1} = 1$ frame that allows the transmission of v_4 frame. The first service round ends with the transmission of v_4. The frame transmission continues in a similar manner in the following rounds.

From this example the operation of WRR scheduling seems similar to that of DRR scheduling presented in Section 2.3.1. However, since WRR assigns credit in terms of the number of frames per class and does not consider the size of these frames the bandwidth utilised by each class may show significant variations. Let us illustrate how the bandwidth is shared among classes in a WRR scheduler in the following paragraphs.

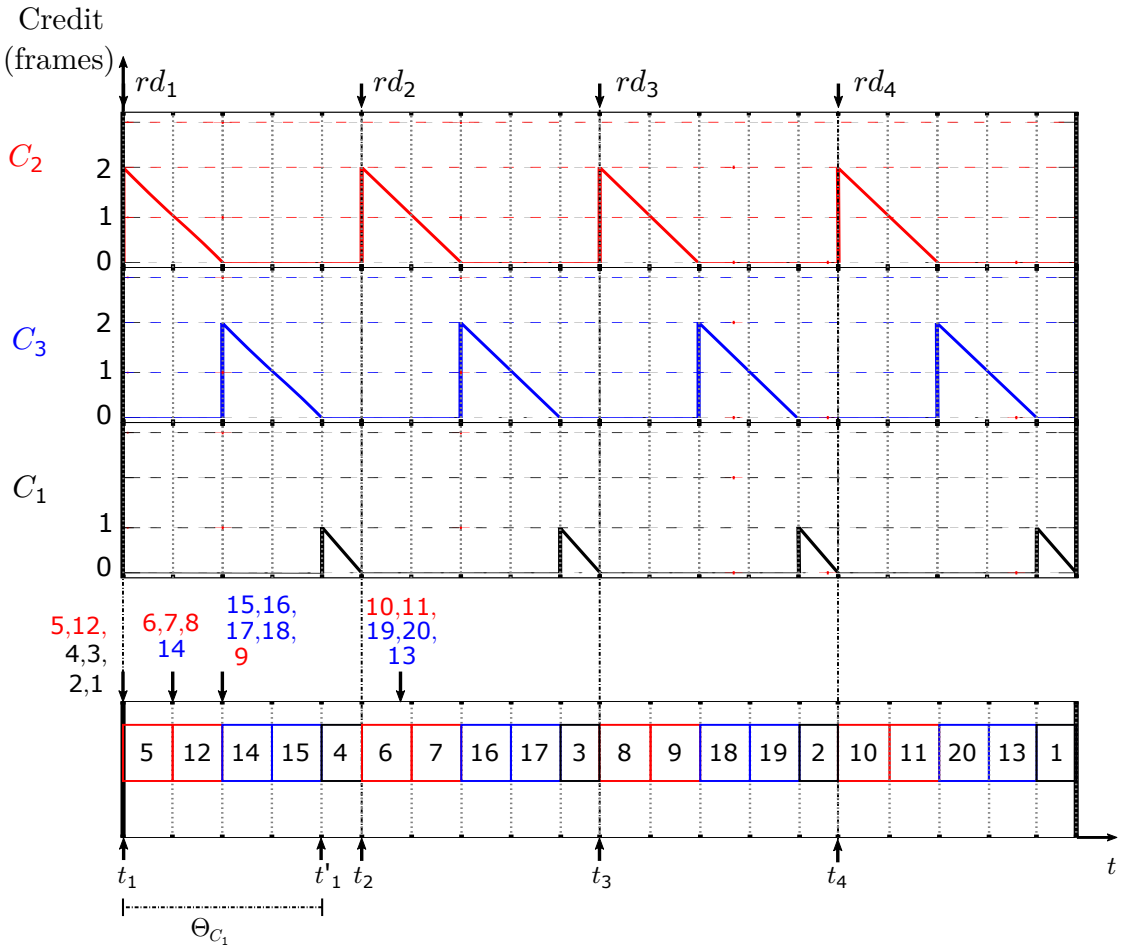


Figure 6.2 WRR scheduling rounds at output port (Figure 6.1)

6.2.2 Bandwidth sharing in WRR scheduler

A WRR scheduler allows managing the bandwidth available to different classes based on the weights assigned to these classes. So, we must assign a weight W_{C_x} to C_x such that it leads to a good approximation of the percentage ϕ_{C_x} of bandwidth R envisioned for C_x such that

$$\sum_{1 \leq x \leq n} \phi^h_{C_x} = 100\%$$

where n is the number of classes served at h.

Indeed the average share of bandwidth actually available to C_x in a given scheduling round also depends upon the size of frames transmitted in this round. Since a class buffer may have frames of different size the total amount of data transmitted in each round can

be different even if the number of frames transmitted in these rounds is the same. This makes it very difficult to guarantee that C_x will receive atleast $\phi_{C_x}\%$ of bandwidth. The average service received by C_x between two consecutive opportunities can be given by

$$\rho_{C_x} = \frac{N_{C_x}}{\sum\limits_{1 \leq j \leq n} N_{C_j}} \times R$$

where N_{C_j} is the number of bytes served from C_j and is equal to the sum of frame lengths served from C_j.

In order to show the bandwidth distribution among classes, let us consider an example of 3 classes (C_1, C_2 and C_3) and analyse the following two scenarios:

case 1: In this case we assume that all the classes are active and they have flows with unique frame size $l_{C_1} = l_{C_2} = l_{C_3} = 100$ bytes (= 800 bits). We also assume that out of total bandwidth $R = 100$ bits/μsec the bandwidth envisioned for C_2 and C_3 is 40 bits/μsec each and that for C_1 is 20 bits/μsec, i.e. $\phi_{C_1} = 20\%$, $\phi_{C_2} = 40\%$ and $\phi_{C_3} = 40\%$. In this case, based on the size of frames that each class can transmit, the classes C_2 and C_3 must be allowed to transmit twice the number of frames transmitted by C_1. If $W_{C_1} = 1$ frame then $W_{C_2} = W_{C_3} = 2$ frames, so that each class gets a bandwidth not less than

$$\begin{aligned}
\rho_{C_1} &= \frac{(W_{C_1} \times l_{C_1})}{(W_{C_1} \times l_{C_1}) + (W_{C_2} \times l_{C_2}) + (W_{C_3} \times l_{C_3})} \times R \\
&= \frac{1 \times 800}{(1 \times 800) + (2 \times 800) + (2 \times 800)} \times 100 = 20\ bits/\mu sec \\
\rho_{C_2} &= \frac{(W_{C_2} \times l_{C_2})}{(W_{C_1} \times l_{C_1}) + (W_{C_2} \times l_{C_2}) + (W_{C_3} \times l_{C_3})} \times R \\
&= \frac{2 \times 800}{(1 \times 800) + (2 \times 800) + (2 \times 800)} \times 100 = 40\ bits/\mu sec \\
\rho_{C_3} &= \frac{(W_{C_3} \times l_{C_3})}{(W_{C_1} \times l_{C_1}) + (W_{C_2} \times l_{C_2}) + (W_{C_3} \times l_{C_3})} \times R \\
&= \frac{2 \times 800}{(1 \times 800) + (2 \times 800) + (2 \times 800)} \times 100 = 40\ bits/\mu sec
\end{aligned}$$

case 2: Let us now assume that the same classes have flows with a range of frame sizes : $l_{C_1}^{min} = 190$ bytes and $l_{C_1}^{max} = 210$ bytes, $l_{C_2}^{min} = 90$ bytes and $l_{C_2}^{max} = 110$ bytes, and $l_{C_3}^{min} = 90$ bytes and $l_{C_3}^{max} = 110$ bytes. In this case, the average bandwidth available to each class can not be fixed by selecting the weights since each class can transmit frames of any length between its minimum and maximum values. For instance, if each class has some

frames to be transmitted in its buffer and the weights assigned to these classes are $W_{C_1} = 1$ and $W_{C_2} = W_{C_3} = 2$, then the bandwidth received by C_1 can vary between 30.15 bits/μsec (when C_1 transmits only $l_{C_1}^{min}$ size frames and other classes transmit only $l_{C_2}^{max}$ and $l_{C_3}^{max}$ size frames) and 36.84 bits/μsec (when C_1 transmits only $l_{C_1}^{max}$ size frames and other classes transmit only $l_{C_2}^{min}$ and $l_{C_3}^{min}$ size frames). Besides, unlike in case 1, in this case, it is not possible to guarantee the bandwidth of at least 20, 40 and 40 bits/μsec by any combination of weights W_{C_1}, W_{C_2} and W_{C_3} for classes C_1, C_2 and C_3 respectively.

The traditional solution is to assign weights such that it approximate the bandwidth near ϕ_{C_x} to the frames of average length $l_{C_x}^{avg}$ in C_x. Which means, in each class, the frames of average lengths

$$\begin{aligned} l_{C_1}^{avg} &= \frac{190 + 210}{2} = 200 \; bytes, \\ l_{C_2}^{avg} &= \frac{90 + 110}{2} = 100 \; bytes, \\ l_{C_3}^{avg} &= \frac{90 + 110}{2} = 100 \; bytes \end{aligned}$$

should get $\phi_{C_1} = 20\%$, $\phi_{C_2} = 40\%$ and $\phi_{C_3} = 40\%$ of $R = 100$ bits/μsec. So, the portion of C_x frames served in each round is $\nabla_{C_x} = \frac{\phi_{C_x}}{l_{C_x}^{avg}}$, i.e.

$$\nabla_{C_1} = \frac{\phi_{C_1}}{l_{C_1}^{avg}} = \frac{20}{200} = 0.1, \; \nabla_{C_2} = \frac{\phi_{C_2}}{l_{C_2}^{avg}} = \frac{40}{100} = 0.4, \; \nabla_{C_3} = \frac{\phi_{C_3}}{l_{C_3}^{avg}} = \frac{40}{100} = 0.4$$

Since the weight should be assigned as a whole number, it is given by the mean number of C_x frames of length $l_{C_x}^{avg}$ as

$$W_{C_x} = round\left(\frac{\nabla_{C_x}}{\min\limits_{1 \le i \le n} \nabla_{C_i}}\right) \tag{6.1}$$

where $round(a)$ gives the nearest integer value to a. Thus, we have,

$$\begin{aligned} W_{C_1} &= round\left(\frac{0.1}{0.1}\right) = 1 \; frame, \\ W_{C_2} &= round\left(\frac{0.4}{0.1}\right) = 4 \; frames, \\ W_{C_3} &= round\left(\frac{0.4}{0.1}\right) = 4 \; frames, \end{aligned}$$

Therefore, when each class has frames of average length in the buffer, they get bandwidths

no less than

$$\rho_{C_1} = \frac{(W_{C_1} \times l_{C_1}^{avg})}{(W_{C_1} \times l_{C_1}^{avg}) + (W_{C_2} \times l_{C_2}^{avg}) + (W_{C_3} \times l_{C_3}^{avg})} \times R$$
$$= \frac{1 \times 1600}{(1 \times 1600) + (4 \times 800) + (4 \times 800)} \times 100 = 20\ bits/\mu sec$$
$$\rho_{C_2} = \frac{(W_{C_2} \times l_{C_2}^{avg})}{(W_{C_1} \times l_{C_1}^{avg}) + (W_{C_2} \times l_{C_2}^{avg}) + (W_{C_3} \times l_{C_3}^{avg})} \times R$$
$$= \frac{4 \times 800}{(1 \times 1600) + (4 \times 800) + (4 \times 800)} \times 100 = 40\ bits/\mu sec$$
$$\rho_{C_3} = \frac{(W_{C_3} \times l_{C_3}^{avg})}{(W_{C_1} \times l_{C_1}^{avg}) + (W_{C_2} \times l_{C_2}^{avg}) + (W_{C_3} \times l_{C_3}^{avg})} \times R$$
$$= \frac{4 \times 800}{(1 \times 1600) + (4 \times 800) + (4 \times 800)} \times 100 = 40\ bits/\mu sec$$

However, since the frames in any class buffer are not always of average size, the minimum bandwidth received by a class can be much less than what is computed above. Indeed, in the worst case a class C_x can have frames of only minimum length $l_{C_x}^{min}$ in its buffer and the competing classes C_y (y≠x) can have frames of only maximum length $l_{C_y}^{max}$, this leads to a minimum bandwidth received by C_x as

$$\rho_{C_x}^{min} = \frac{(W_{C_x} \times l_{C_x}^{min})}{(W_{C_x} \times l_{C_x}^{min}) + \sum_{\substack{i=1 \\ i \neq x}}^{n} (W_{C_i} \times l_{C_i}^{max})} \times R \tag{6.2}$$

Therefore, in the long term C_1, C_2 and C_3 can receive bandwidths no less than

$$\rho_{C_1}^{min} = \frac{(W_{C_1} \times l_{C_1}^{min})}{(W_{C_1} \times l_{C_1}^{min}) + (W_{C_2} \times l_{C_2}^{max}) + (W_{C_3} \times l_{C_3}^{max})} \times R$$
$$= \frac{1 \times 1520}{(1 \times 1520) + (4 \times 880) + (4 \times 880)} \times 100 = 17.75\ bits/\mu sec$$
$$\rho_{C_2}^{min} = \frac{(W_{C_2} \times l_{C_2}^{min})}{(W_{C_1} \times l_{C_1}^{max}) + (W_{C_2} \times l_{C_2}^{min}) + (W_{C_3} \times l_{C_3}^{max})} \times R$$
$$= \frac{4 \times 720}{(1 \times 1680) + (4 \times 720) + (4 \times 880)} \times 100 = 35.64\ bits/\mu sec$$
$$\rho_{C_3}^{min} = \frac{(W_{C_3} \times l_{C_3}^{min})}{(W_{C_1} \times l_{C_1}^{max}) + (W_{C_2} \times l_{C_2}^{max}) + (W_{C_3} \times l_{C_3}^{min})} \times R$$
$$= \frac{4 \times 720}{(1 \times 1680) + (4 \times 880) + (4 \times 720)} \times 100 = 35.64\ bits/\mu sec$$

The difference between the minimum bandwidth that can be provided to the classes and the bandwidth envisioned for them is the major drawback of WRR scheduling. This difference increases further when the range of frame lengths in each class is increased, however, it becomes less evident when the range of frame lengths is reduced (as in case 1). Besides, WRR introduces a small scheduler latency, which is discussed in the following paragraphs.

6.3 Worst-Case Traversal Time analysis

As shown in previous chapters the aim of worst-case delay analysis is to upper bound the delay for each flow in the network. This delay upper bound corresponds to the minimum service received by a given flow in the presence of maximum traffic in the network. In NC approach, the minimum residual service available to any class flows is modelled as a latency-rate ($\mathcal{LR}$–) server which considers that these flows initially experience maximum scheduler latency and then they are served at minimum service rate until the end of service. Thus, in order to compute the delay upper bound using the NC approach, it is important to compute the maximum latency in WRR scheduler.

In the following paragraphs, we first present the WRR scheduler latency and then we show how our optimised NC approach can be extended for WCTT analysis in WRR scheduling based switched Ethernet network.

6.3.1 WRR scheduler latency

The WRR scheduler latency ($\Theta^h_{C_x}$) is the maximum delay experienced by a class C_x flows, before being served for the first time in its active period. This maximum delay corresponds to the worst-case scenario where C_x becomes active (with the arrival of the first frame in C_x queue) at an instant when it just misses its turn to be served and it has to wait while the maximum possible traffic is served from the competing classes (i.e. they consume their maximum credit in transmission of maximum size frames). Thus, in a WRR scheduler serving n traffic classes at an output port h, the latency experienced by a class C_x flows is given by:

$$\Theta^h_{C_x} = \frac{\sum_{j=1, j\neq x}^{n} (W^h_{C_j} \times l^{max,h}_{C_j})}{R} \tag{6.3}$$

At the switch output port shown in Figure 6.1, the scheduler latency for C_1 flows can be illustrated through the frame sequence shown in Figure 6.2. In this example, the

weights assigned to each class are $W_{C_1} = 1$, $W_{C_2} = 2$ and $W_{C_3} = 2$ frames. Based on the frame sequence in Figure 6.2, C_1 becomes active at t_1, with the arrival of v_4, v_3, v_2 and v_1, and it has to wait until t'_1 for the first service while C_2 and C_3 consume their maximum credit in transmission of v_5, v_{12}, v_{14} and v_{15} frames. If the frames served from C_2 and C_3 are of maximum length then we have

$$t'_1 - t_1 = \frac{(100 + 100 + 100 + 100) \times 8}{100} = 32\ \mu sec$$

which is equal to the maximum scheduler latency given by equation 6.3 as

$$\Theta_{C_1} = \frac{(W_{C_2} \times l_{C_2}^{max}) + (W_{C_3} \times l_{C_3}^{max})}{R} = \frac{(2 \times 100 \times 8) + (2 \times 100 \times 8)}{100} = 32\ \mu sec$$

Unlike in DRR scheduling, where a class flows can experience scheduler latency as multiple delays in first two rounds because they may receive reduced service (see Section 2.4.1), the scheduler latency in WRR scheduling is considered only in the first round. Indeed, in DRR scheduler each class is assigned credit in terms of number of bytes (quantum) and in the worst case a class can transmit data less than its assigned quantum leaving a deficit to be utilised in the next round while other classes transmit data equal to their assigned quantum which leads to a reduced service in the given round whereas in WRR scheduler each class is assigned a credit in terms of fixed number of frames (weight) and these classes can transmit frames exactly equal to their allocated weights in any round. Thus, in WRR scheduling the flows are always served at the average service rate not less than the minimum service rate (equation 6.2) in each scheduling round.

In the given example, the minimum service rate for C_1 flows is

$$\begin{aligned}\rho_{C_1}^{min} &= \frac{(W_{C_1} \times l_{C_1}^{min})}{(W_{C_1} \times l_{C_1}^{min}) + (W_{C_2} \times l_{C_2}^{max}) + (W_{C_3} \times l_{C_3}^{max})} \times R \\ &= \frac{1 \times 640}{(1 \times 640) + (2 \times 800) + (2 \times 800)} \times 100 = 16.667\ bits/\mu sec\end{aligned}$$

6.3.2 Network Calculus approach

The modelling of WRR scheduling based network in NC approach is similar to the one described for DRR scheduling based network in Chapter 4, the only difference is in the service model since these two scheduling policies adapt different mechanism of credit assignment. In the following paragraphs, we show how the NC approach can be applied in the context of WRR scheduling based network.

According to the NC approach, the worst-case delay of class C_x flows at a switch

output port h is determined by the maximum horizontal distance between its overall arrival curve α_{C_x} and the service curve β_{C_x} as

$$h(\alpha^h_{C_x}, \beta^h_{C_x}) = \sup_{s \geq 0}(\inf\{\tau \geq 0 | \alpha^h_{C_x}(s) \leq \beta^h_{C_x}(s+\tau)\}) \tag{6.4}$$

In order to consider the worst-case scenario α_{C_x} considers the maximum traffic in C_x, which corresponds to cumulative traffic of C_x flows arriving from all the inputs at h, and β_{C_x} considers the minimum service provided to C_x flows, which corresponds to the case when these flows experience maximum scheduler latency and minimum service rate.

Let us illustrate the computation of this delay on the network example shown in Figure 6.1. For that purpose let us focus on flow v_1 of class C_1.

In the given network, the cumulative traffic of C_1 flows is the sum of individual arrival curves of flows v_1, v_2, v_3 and v_4, we have

$$\alpha_{C_1} = \alpha_{v_1} + \alpha_{v_2} + \alpha_{v_3} + \alpha_{v_4}$$

where,

$$\begin{aligned}
\alpha_{v_1} &== 0.0061\ t + 792.1 \\
\alpha_{v_2} &== 0.0125\ t + 800.3 \\
\alpha_{v_3} &== 0.0123\ t + 792.3 \\
\alpha_{v_4} &== 0.0125\ t + 800.3
\end{aligned}$$

so,

$$\alpha_{C_1} = 0.0434\ t + 3185$$

Since the C_1 flows compete with C_2 and C_3 flows at the given output port and the WRR scheduler serves these flows based on the weight assigned to each class, the C_1 flows receive minimum service when they experience the maximum scheduler latency (equation (6.3)) and minimum service rate (equation (6.2)). Thus according to NC approach, the residual service for C_1 flows can be given by a service curve (shown in Figure 6.3) as

$$\beta_{C_1} = \rho^{min}_{C_1}[t - \Theta_{C_1}]^+ = 16.667\ [t - 32]^+ \tag{6.5}$$

However, we have seen in the case of DRR scheduling (Section 4.2.2) that this model of service curve can be pessimistic as it does not take into account the exact composition

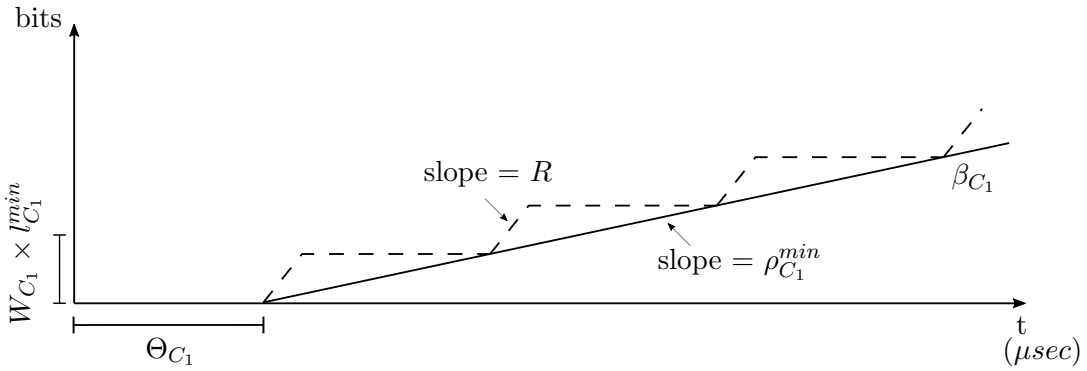


Figure 6.3 NC service curve for C_1 with WRR scheduling

of each class. Indeed, it can be the case for WRR scheduling as well. In WRR scheduling, the computation of $\rho^{min}_{C_x}$ and Θ_{C_x} is based on the assumption that during the active period of C_x the competing classes C_y (y≠x) are always active and they serve at least W_{C_y} frames of $l^{max}_{C_y}$ bytes in each round. This assumption is correct only when C_y has enough frames of maximum size in its buffer during the active period of C_x, otherwise, β_{C_y} can be pessimistic. For example, in the WRR scheduling rounds shown in Figure 6.2, since there are enough frames in C_2 and C_3, the C_1 frames are delayed by them in each round. But, let us assume that C_2 had only one flow v_5 and C_3 had only one flow v_{14}, in that case, C_1 flows would experience a latency at most

$$t'_1 - t_1 = \frac{(100 + 100) \times 8}{100} = 16\ \mu sec$$

which is much smaller than the computed WRR scheduler latency $\Theta_{C_1} = 32\ \mu$sec. Moreover, the C_1 flows would be served much earlier as there are no frames in C_2 and C_3 in the following rounds, as shown in Figure 6.4.

The scenario shown in Figure 6.4 may not be the worst-case one, but in any case, with reduced traffic C_2 and C_3 cannot delay C_1 flows in each scheduling round. Thus, the delay computation by equation (6.4), which is based on β_{C_1}, can be pessimistic. The delay upper bound for v_1 from equation (6.4) is

$$D_1 = h(\alpha_{C_1}, \beta_{C_1}) = \sup_{s \geq 0}(\inf\{\tau \geq 0 | \alpha_{C_1}(s) \leq \beta_{C_1}(s + \tau)\}) = 223.1\ \mu sec$$

The optimisation proposed in the context of DRR scheduling in Section 4.3 aims to mitigate the pessimism by removing the over-estimated portion of competing class traffic considered by service curve in the delay computation. This over-estimated traffic is obtained from the difference between the service load SL_{C_y}, which represents the

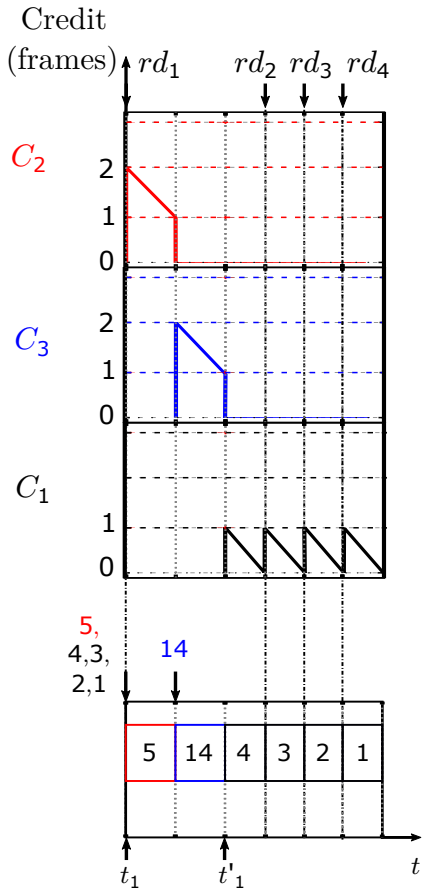


Figure 6.4 C_1 service with reduce traffic from C_2 and C_3

traffic of a competing class C_y considered by service curve β_{C_x} in computation of delay bound D_i, and the effective maximum load $L_{C_y}^{max}$, which represents the upper bound on the actual traffic in C_y during the active period [0, D_i] of C_x. If $SL_{C_y} > L_{C_y}$, then the difference SL_{C_y} - L_{C_y} represent the over-estimated portion of C_y in β_{C_x} which can be safely removed in the delay computation. In the following paragraphs, we show how this optimisation can be applied in the context of WRR scheduling. The main difference is in the computation of service load SL_{C_y} since the WRR scheduling rounds are different from that of DRR scheduling.

The optimisation of delay upper bound $D_1 = 223.1$ μsec computed for v_1 proceeds as follows:

- Compute the maximised traffic of C_2 considered by β_{C_1} in interval $[0, D_1]$. This

maximum traffic is given by service load $SL_{C_2}(D_1)$.

The service of competing class C_2, considered by a service curve β_{C_1} in equation (6.5) (also shown in Figure 6.3), can be described in two parts. First part is the interval $(0, \Theta_{C_1}]$ that corresponds to the scheduler latency Θ_{C_1}. In this interval C_1 does not receive any service while C_2 receives a service of $W_{C_2} \times l_{C_2}^{max}$ bytes. Second part is the interval $(\Theta_{C_1}, +inf)$ which is divided into multiple identical intervals of length t_N, where in each interval C_1 receives the minimum service of $W_{C_1} \times l_{C_1}^{min}$ and C_2 gets a service of at most $W_{C_2} \times l_{C_2}^{max}$ bytes.

These intervals in service curve can be generalised for a WRR scheduler serving any number of classes (n) at an output port h. In the service curve $\beta_{C_x}^h$ of a class C_x, the lower bound on the maximised traffic of competing class C_y can be given by service load $SL_{C_y}^h(\mathrm{t})$ as:

$$SL_{C_y}^h(t) = \begin{cases} 0, & t < \Theta_{C_x}^h \\ (W_{C_y}^h \times l_{C_y}^{max,h}) \left(1 + \left\lfloor \dfrac{(t - \Theta_{C_x}^h)}{t_N} \right\rfloor\right), & \Theta_{C_x}^h \leq t \end{cases} \tag{6.6}$$

where,

$$t_N = \frac{(W_{C_x}^h \times l_{C_x}^{min,h})}{R} + \frac{\sum_{j=1, j\neq x}^{n^h} (W_{C_j}^h \times l_{C_j}^{max,h})}{R}$$

In any interval the corresponding load is taken into account by $SL_{C_y}^h(\mathrm{t})$ only at the end of this interval. For instance, since the competing class C_y is served $(W_{C_y}^h \times l_{C_y}^{max,h})$ bytes in interval $(0, \Theta_{C_x}^h]$, the load of C_y before $\Theta_{C_x}^h$ is 0. The $(W_{C_y}^h \times l_{C_y}^{max,h})$ bytes from C_y are considered at $t = \Theta_{C_x}^h$, this load remains unchanged until the next service of C_y. In the following intervals, the load of $(W_{C_y}^h \times l_{C_y}^{max,h})$ bytes is accumulated at the end of each interval. Thus, $SL_{C_y}^h(\mathrm{t})$ is a lower-bound on the maximized service considered by $\beta_{C_x}^h$. This service load is also shown in Figure 6.5.

For v_1, the service load from competing class C_2 in the interval $[0, D_1]$ can be computed as:

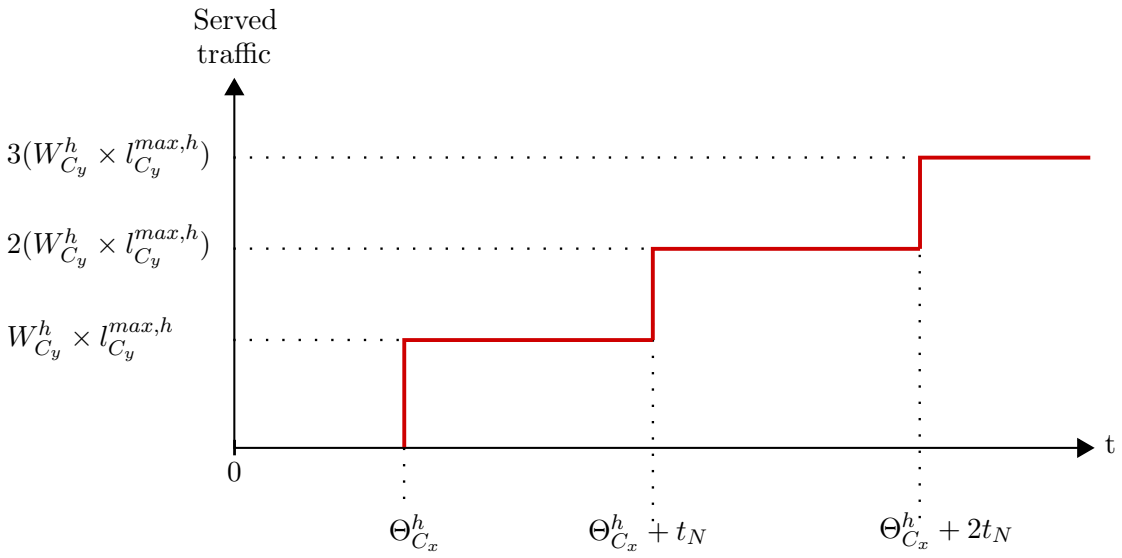


Figure 6.5 Lower bound on maximum service load in WRR scheduler

We have,

$$\begin{aligned} D_1 &= 223.1\mu sec \\ \Theta_{C_1} &= 32\mu sec \\ t_N &= \frac{(W_{C_1} \times l^{min}_{C_1})}{R} + \frac{(W_{C_2} \times l^{max}_{C_2}) + (W_{C_3} \times l^{max}_{C_3})}{R} \\ &= \frac{(1 \times 640)}{100} + \frac{(2 \times 800) + (2 \times 800)}{100} = 38.4\mu sec \end{aligned}$$

Since, $D_1 > \Theta_{C_1}$, *therefore*

$$\begin{aligned} SL_{C_2}(D_1) &= (W_{C_2} \times l^{max}_{C_2}) \left(1 + \left\lfloor \frac{(D_1 - \Theta_{C_1})}{t_N} \right\rfloor\right) \\ &= (2 \times 800) \left(1 + \left\lfloor \frac{(223.1 - 32)}{38.4} \right\rfloor\right) = 8000 \ bits \ = 1000 \ bytes \end{aligned}$$

- Compute the upper bound on actual traffic in C_2 in interval $[0, D_1]$. This upper bound is given by effective maximum load $L^{max}_{C_2}(D_1)$, which is equal to the overall arrival curve of C_2 flows in interval $[0, D_1]$.

  At the given switch output port, the traffic in C_2 is composed of 8 flows: v_5, v_6, v_7, v_8, v_9, v_{10}, v_{11} and v_{12}. The overall arrival $\alpha_{C_2}(t)$ can be obtained by the sum

of individual arrival curves of each flow in C_2. However, the flows v_7, v_9 and v_{10} as well as the flows v_8, v_{11} and v_{12} are serialised on their shared input link (this serialization effect is explained in Section 1.5.2) and thus they can be represented by a single arrival curve respectively. We have,

$$\begin{aligned}\alpha_{C_2}(t) = \ & \alpha_{v_5} + \alpha_{v_6} \\ & + \min\{((R \times t) + \max\{b_{v_7}, b_{v_9}, b_{v_{10}}\}), (\alpha_{v_7} + \alpha_{v_9} + \alpha_{v_{10}})\} \\ & + \min\{((R \times t) + \max\{b_{v_8}, b_{v_{11}}, b_{v_{12}}\}), (\alpha_{v_8} + \alpha_{v_{11}} + \alpha_{v_{12}})\}\end{aligned}$$

where, $\alpha_{v_5} = 0.00625\ t + 800$, $\alpha_{v_6} = 0.00625\ t + 800.15$, $\alpha_{v_7} = 0.0061\ t + 792.15$, $\alpha_{v_8} = 0.0061\ t + 792.15$, $\alpha_{v_9} = 0.0061\ t + 792.15$, $\alpha_{v_{10}} = 0.0125\ t + 800.3$, $\alpha_{v_{11}} = 0.0123\ t + 792.3$ and $\alpha_{v_{12}} = 0.00625\ t + 800.15$

Therefore, the effective load of C_2 in interval $[0, D_{C_1}]$ is

$$\begin{aligned}L^{max}_{C_2}(D_{C_1}) &= \alpha_{C_2}(D_{C_1}) \\ &= \alpha_{C_2}(223.1) \\ &= 6383\ bits = 797.875\ bytes\end{aligned} \tag{6.7}$$

- Compute the over-estimated portion of C_2 in β_{C_1}. If $SL_{C_2}(D_1) > L^{max}_{C_2}(D_1)$ then the difference between the two represents the over-estimated portion of C_2 traffic in β_{C_1}. Otherwise, the difference is assumed to be 0.

$$\max\{SL_{C_2}(D_1) - L^{max}_{C_2}(D_1), 0\} = 1000 - 797.875 = 202.125\ bytes$$

- Repeat the same process for the next competing class C_3, to get

$$\begin{aligned}SL_{C_3}(D_1) &= (W_{C_3} \times l^{max}_{C_3}) \left(1 + \left\lfloor \frac{(D_1 - \Theta_{C_1})}{t_N} \right\rfloor\right) \\ &= (2 \times 800) \left(1 + \left\lfloor \frac{(223.1 - 32)}{38.4} \right\rfloor\right) = 8000\ bits = 1000\ bytes \\ L^{max}_{C_3}(D_{C_1}) &= \alpha_{C_3}(D_{C_1}) \\ &= \alpha_{C_3}(223.1) = 6389\ bits = 798.625\ bytes\end{aligned}$$

Therefore, the over-estimated portion of C_3 in β_{C_1} is

$$\max\{(SL_{C_3}(D_1) - L^{max}_{C_3}(D_1), 0\} = 1000 - 798.625 = 201.375\ bytes$$

- Finally, the delay considered in transmission of the overestimated traffic of ($SL_{C_2}(D_1)$ $-L_{C_2}^{max}(D_1)$) bytes and ($SL_{C_3}(D_1) - L_{C_3}^{max}(D_1)$) bytes at link rate ($R = 100$ bits/μsec) can be removed from the delay upper bound D_1 in order to get the optimised delay $D_{1,opt}$.

$$D_{1,opt} = D_1 - \left(\frac{\max\{SL_{C_2}(D_1) - L_{C_2}^{max}(D_1\}}{R} + \frac{\max\{SL_{C_3}(D_1) - L_{C_3}^{max}(D_1\}}{R} \right)$$
$$= 223.1 - \frac{(202.125 \times 8)}{100} - \frac{201.375 \times 8}{100} = 190.82\ \mu sec$$

6.4 Case study on industrial AFDX network

In this section we perform a case study on an industrial size AFDX network configuration. Our main aim is to perform WCTT analysis on this network under WRR scheduling and compare the results with DRR scheduling. First, we show that since WRR scheduling does not take into account the size of the frames served from different traffic classes it can lead to significantly unfair bandwidth distribution among these classes. Moreover, in case of an existing industrial AFDX configuration (presented earlier in Section 5.2.2) having flows with a wide variety of frame lengths, it leads to a scenario where WCTT analysis based on NC approach cannot be successfully performed.

Then, we present another industrial size AFDX network configuration (having flows with a smaller variety of frame lengths) in order to perform WCTT analysis under WRR scheduling and compare the results to those obtained under DRR scheduling.

6.4.1 WCTT analysis on an industrial size AFDX network configuration

Let us consider the industrial configuration of AFDX network presented in the previous chapter (Section 5.2.2). It has 8 switches forwarding 984 flows (a.k.a. Virtual Link (VL)) on a total of 6276 paths. The link rate is $R = 100$ bits/μsec. The flow dispatching among BAGs and maximum frame lengths is recalled in Table 6.2.

The flows are arbitrarily distributed between 2 critical classes: a class C_1 for critical flows and a class C_2 for less-critical flows. This distribution is based on the fact that the critical flows with low BAG values can have smaller delay constraints as compared to the flows with larger BAG. Thus, all the flows with BAG values up to 16 msec are considered in critical class C_1 (having deadline 12573 μsec) and the remaining flows are considered

Table 6.2 Reacall of flow characteristics in industrial configuration (a) BAG (b) Frame length

BAG (ms)	No. of VLs
2	20
4	40
8	78
16	142
32	229
64	220
128	255

(a)

Frame Length (bytes)	No. of VLs
84-200	278
201-400	396
401-600	157
601-900	69
901-1200	28
1201-1500	51
> 1500	5

(b)

in less-critical class C_2 (having deadline 50292 μsec). Table 6.3 summarises the features of these classes.

Table 6.3 Flow differentiation in industrial configuration

Flow count	BAG Range ($msec$)	Frame length range ($bytes$)	Class	Deadline (μsec)
280 (1681 paths)	2 – 16	84 – 1497	Critical C_1	12573
704 (4595 paths)	32 – 128	84 – 1535	Less-Critical C_2	50292
Additional flows				
10 (501 paths)	4 – 8	84 – 934	Best-Effort C_3	-

In the given configuration, we have arbitrarily introduced 10 additional flows which share the path with each critical flow on at least one switch output port. The added flows are considered as non-critical flows characterised by class C_3 shown in Table 6.3. Since C_3 flows share the path with critical flows they can have an impact on the worst-case end-to-end delay of critical flows. Our aim is to analyse this delay when the switch output port employs the WRR scheduling policy and compare it with DRR scheduling.

We have seen in the previous chapter that in case of DRR scheduling a quantum assignment can be achieved, for the given network configuration, with the help of

Algorithm 2 that leads to minimum delay in non-critical flows (C_3) while allowing the critical flows (C_1 and C_2) to respect their deadlines. It belongs to the distribution of quanta $Q = 6340$ bytes as $Q_{C_1} = 3659$ bytes, $Q_{C_2} = 1535$ bytes and $Q_{C_3} = 1146$ bytes. In this case, the minimum bandwidth available to each class is

$$\rho_{C_1}^{DRR} = 57.71\ bits/\mu sec,\ \rho_{C_2}^{DRR} = 24.21\ bits/\mu sec,\ \rho_{C_3}^{DRR} = 18.07\ bits/\mu sec$$

In order to perform a fair comparison between DRR and WRR scheduling, we must achieve a scenario which is similar (if not same) in the two schedulers. In WRR scheduling, a guarantee on minimum bandwidth close to that of DRR scheduling may not be achieved easily. This is due to the fact that the WRR scheduler serves each class based on a predefined weights (i.e. a fixed number of frames) irrespective of the frame size and, as shown in Section 6.2.2, the bandwidth available to any class depends not only on the weights assigned to each class but also on the range of frame sizes in each class. Indeed, the service based only on the number of frames is not fair since one class may transmit frames of only minimum size while other class may transmit frames of only maximum size.

This problem can be shown on the given industrial configuration. Let us assume that each class C_x (x = 1, 2 and 3) is expected to have a bandwidth share close the one provided in DRR scheduling. Let us verify if it is possible to achieve such bandwidth distribution for these classes. From equation (6.2) we have

$$\rho_{C_x}^{min} = \frac{(W_{C_x} \times l_{C_x}^{min})}{(W_{C_x} \times l_{C_x}^{min}) + \sum_{y \neq x} (W_{C_y} \times l_{C_y}^{max})} \times R$$

so,

$$\rho_{C_1}^{min} = \frac{(W_{C_1} \times 84)}{(W_{C_1} \times 84) + (W_{C_2} \times 1535) + (W_{C_1} \times 934)} \times 100$$
$$\rho_{C_2}^{min} = \frac{(W_{C_2} \times 84)}{(W_{C_1} \times 1497) + (W_{C_2} \times 84) + (W_{C_1} \times 934)} \times 100$$
$$\rho_{C_3}^{min} = \frac{(W_{C_3} \times 84)}{(W_{C_1} \times 1497) + (W_{C_2} \times 1535) + (W_{C_1} \times 84)} \times 100$$

Instead of directly looking for the value of weights that can lead to the bandwidths 57.71, 24.21 and 18.07 bits/μsec for C_1, C_2 and C_3 respectively, let us first assume some arbitrary weights and check the minimum bandwidths available to each class, so let us

assume, $W_{C_1} = W_{C_2} = W_{C_3} = 1$ frame, we get

$$\rho_{C_1}^{min} = 3.29\ bits/\mu sec$$
$$\rho_{C_2}^{min} = 3.33\ bits/\mu sec$$
$$\rho_{C_3}^{min} = 2.69\ bits/\mu sec$$

which are far from the desired minimum bandwidths. Since all the classes have bandwidths less than their expected values and we have assigned a minimum possible weight (= 1 frame) to each class, we must increase the assigned weights in order to increase their bandwidth share. However, it must be noticed that in the computation of minimum bandwidth, the increase in weight W_{C_x} assigned to a class C_x affects the bandwidth of C_x in proportion to its minimum frame length ($l_{C_x}^{min}$) and affects the bandwidth of other classes C_y (y≠x) in proportion to the maximum frame length ($l_{C_x}^{max}$) of C_x. Which means, when the difference between the $l_{C_x}^{min}$ and $l_{C_x}^{max}$ is large, the increase in W_{C_x} shall result in a disproportionate effect on change in bandwidth for all the classes. Thus, in this case, it is not possible to increase the bandwidth share of any class without decreasing that of any other class. Therefore, the minimum bandwidth share of all the classes cannot be increased together to the values near their expected values. This makes it impossible to perform a fair comparison between WRR and DRR scheduling for the given industrial configuration.

Another problem in case of WRR scheduling in WCTT analysis of the given network configuration is that the minimum bandwidths computed above are too low to be able to compute the delay using the NC approach. In order to compute the delay by equation (6.4) the service curve and arrival curve must converge but in the given network configuration they do not converge since the maximum long term arrival rate (i.e. the slope of arrival curve) at several switch output ports is much larger than the slope of service curve (i.e. $\rho_{C_1}^{min} = 3.29$, $\rho_{C_2}^{min} = 3.33$ and $\rho_{C_3}^{min} = 2.69$) and hence the delay cannot be computed.

One solution to these problems is to use a different method of flow differentiation. For instance, instead of differentiating the flows based on their BAG values, which is the case in Table 6.3, these flows can be differentiated based on their frame sizes so that each class contains the flows of certain (small) range of frame sizes. Another solution is to break the classes in Table 6.3 into sub-classes so that the difference between the frame sizes within each class is reduced. However, the later solution may not completely resolve the problem as the total bandwidth should be divided among more number of classes leading to a smaller share of the bandwidth available to each class. Both solutions lead to a completely different problem of optimising the distribution of flows either based on

the frame sizes or based on the number of optimal classes. We have not focused on this optimisation problem in this thesis.

Instead, we perform the WCTT analysis on another but similar industrial configuration in the following paragraphs.

6.4.2 WCTT analysis on a modified configuration of an industrial size AFDX network

We consider an AFDX network configuration which is similar to the previous one in terms of switches, flow paths and end-systems. The only difference is that the frame length in each flow is limited to certain upper and lower limit. More precisely, we review two cases, first, we assume critical flows of homogeneous frame length per class, second, we assume critical flows with a small range of frame lengths. In both cases, we study the worst-case end-to-end delays in each class in the presence of non-critical flows in the network.

6.4.2.1 WRR scheduling with homogeneous flows

Let us focus on the network configuration with flows of homogeneous frame length shown in Table 6.4. This configuration is similar to the previous one but the flows in each class are constrained to transmit frames of unique lengths. Actually, the frame length in each class is equal to the average frame length in previous configuration, i.e. $l_{C_1} = 350$ bytes and $l_{C_2} = 400$ bytes. The flow differentiation is unchanged, i.e. critical class C_1 has the flows with BAG values up to 16 msec and the remaining flows are assigned in less-critical class C_2. The arbitrarily introduced non-critical flows are characterised by class C_3. We examine seven different scenarios by varying the number of additional flows between 10 and 70 flows. In each scenario, we compute the worst-case delay by assuming WRR scheduling at each switch output port and compare the results with DRR scheduling.

In DRR scheduling we assign quantum to each class based on our algorithm (Algorithm 2 in Chapter 3) which leads to distribution $Q_{C_1} = 770$ bytes, $Q_{C_2} = 400$ bytes and $Q_{C_3} = 377$ bytes and the minimum bandwidth available to each class is

$$\rho_{C_1}^{DRR} = 49.77\ bits/\mu sec,\ \rho_{C_2}^{DRR} = 25.85\ bits/\mu sec,\ \rho_{C_3}^{DRR} = 24.38\ bits/\mu sec$$

In WRR scheduling an approximately similar bandwidth distribution can be achieved by the traditional weight assignment method shown in Section 6.2.2 as:

Table 6.4 Flow differentiation in industrial configuration with homogeneous flows

Flow count	BAG Range ($msec$)	Frame length range ($bytes$)	Class	Deadline (μsec)
280 (1681 paths)	2 – 16	350	Critical C_1	12573
704 (4595 paths)	32 – 128	400	Less-Critical C_2	50292
Additional flows				
10 – 70 (501 – 699 paths)	4 – 8	350	Best-Effort C_3	-

since,

$$\phi_{C_1} = \frac{\rho_{C_1}^{DRR}}{R} \times 100 = 49.77\%,$$
$$\phi_{C_2} = \frac{\rho_{C_2}^{DRR}}{R} \times 100 = 25.85\%$$
$$\phi_{C_3} = \frac{\rho_{C_3}^{DRR}}{R} \times 100 = 24.38\%$$

$$\nabla_{C_1} = \frac{\phi_{C_1}}{l_{C_1}} = \frac{49.77}{350},\ \nabla_{C_2} = \frac{\phi_{C_2}}{l_{C_2}} = \frac{25.85}{400},\ \nabla_{C_3} = \frac{\phi_{C_3}}{l_{C_3}} = \frac{24.38}{350}$$

we get,

$$W_{C_1} = round\left(\frac{\nabla_{C_1}}{\min\{\nabla_{C_1}, \nabla_{C_2}, \nabla_{C_3}\}}\right) = 2 frames$$
$$W_{C_2} = round\left(\frac{\nabla_{C_2}}{\min\{\nabla_{C_1}, \nabla_{C_2}, \nabla_{C_3}\}}\right) = 1 frame$$
$$W_{C_3} = round\left(\frac{\nabla_{C_3}}{\min\{\nabla_{C_1}, \nabla_{C_2}, \nabla_{C_3}\}}\right) = 1 frame$$

and

$$\rho_{C_1}^{min,WRR} = \frac{(2 \times 350)}{(2 \times 350) + (1 \times 400) + (1 \times 350)} \times 100 = 48.27\ bits/\mu sec$$
$$\rho_{C_2}^{min,WRR} = \frac{(1 \times 400)}{(2 \times 350) + (1 \times 400) + (1 \times 350)} \times 100 = 27.58\ bits/\mu sec$$

$$\rho_{C_3}^{min,WRR} = \frac{(1 \times 350)}{(2 \times 350) + (1 \times 400) + (1 \times 350)} \times 100 = 24.13\ bits/\mu sec$$

The worst-case end-to-end delay comparison in DRR and WRR scheduling is shown in Figure 6.6. Since the number of C_3 flows is increased in each scenario which leads to variation in end-to-end delay for each flow in C_1 and C_2 differently, we only compare the average and maximum delay in both schedulers. In Figure 6.6, in case of 10 C_3 flows the average delays in critical classes C_1 and C_2 is slightly lower in DRR as compared to WRR and on increasing the number of C_3 flows the difference between the two remains more or less same. On the other hand, the delay in non-critical class C_3 is lower in WRR as compared to DRR.

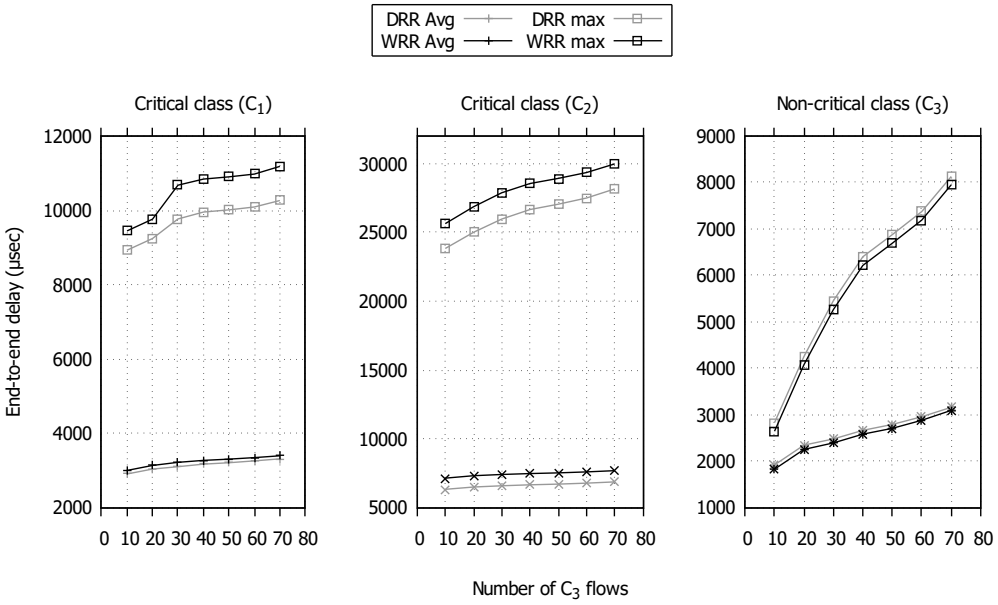


Figure 6.6 End-to-end delay comparison in DRR and WRR with homogeneous flows.

These results are very interesting as they show the difference between the delay output in the two scheduling algorithm under unique conditions. Let us discuss the results for each class separately in the following paragraphs:

Critical class C_1: In case of C_1 flows the delay in DRR is less than that in WRR scheduling. The difference between the delays in the two scheduling can be understood by observing their service curve. The service curve in NC essentially reveals the scheduler latency Θ (which is the worst-case delay due to competing class C_2 and C_3 flows at the beginning of C_1 service) and the service rate ρ (which is the minimum service received

by C_1 flows on the long term) as

$$(\Theta_{C_1}^{WRR} = 60\ \mu sec) < \ (\Theta_{C_1}^{DRR} = 150.17\ \mu sec)$$
$$(\rho_{C_1}^{WRR} = 48.27\ bits/\mu sec) < \ (\rho_{C_1}^{DRR} = 49.77\ bits/\mu sec)$$

Even if the C_1 flows have an advantage of lower latency in WRR, the higher service rate in DRR leads to a favourable condition for these flows on the long term and eventually the delay is less in DRR scheduling.

This scenario is represented in Figure 6.7. The DRR service curve $\beta_{C_1}^{DRR}$ is initially at the right of WRR service curve $\beta_{C_1}^{WRR}$ due to the higher scheduler latency $\Theta_{C_1}^{DRR}$ and since the service rate $\rho_{C_1}^{DRR}$ is higher than $\rho_{C_1}^{WRR}$ the $\beta_{C_1}^{DRR}$ increases rapidly and consequently it is at the left of $\beta_{C_1}^{WRR}$. When the active period of C_1 flows (represented by shaded area) is large enough, the horizontal distance between C_1 arrival curve and service curve (i.e. the delay) is less in DRR.

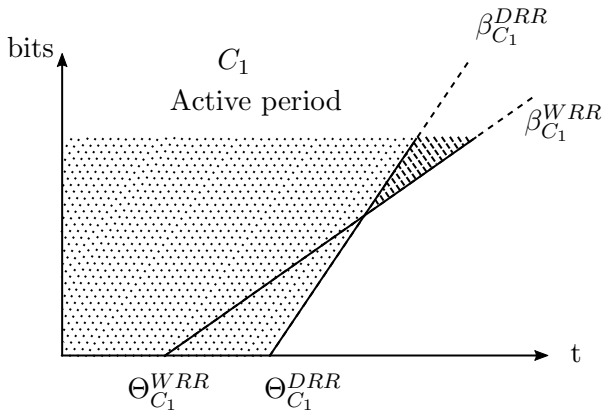


Figure 6.7 Illustration of C_1 flow service

Critical class C_2: In case of C_2 flows, despite the lower scheduler latency and higher service rate in WRR scheduling the delay is higher as compared to DRR.

$$(\Theta_{C_2}^{WRR} = 84\ \mu sec) < \ (\Theta_{C_2}^{DRR} = 147.6\ \mu sec)$$
$$(\rho_{C_2}^{WRR} = 27.58\ bits/\mu sec) > \ (\rho_{C_2}^{DRR} = 25.85\ bits/\mu sec)$$

It is to be noted that $\Theta_{C_2}^{WRR}$, $\Theta_{C_2}^{DRR}$, $\rho_{C_2}^{WRR}$ and $\rho_{C_2}^{DRR}$ are based on pessimistic service curve which does not take into account the effective traffic from competing classes. The delay computation with our optimised NC approach eliminates this pessimism and it

turns out that in case of C_2 flows the delay is less in DRR as compared to WRR, which means that the competing classes (C_1 and C_3) have less effective traffic than what is considered by the service curve and this limited traffic is served earlier in DRR so the actual service received by C_2 flows is better in DRR as compared to WRR.

This scenario is also shown in Figure 6.8. In the worst-case scenario, in WRR scheduling the competing classes C_1 and C_3 can transmit at most

$$\begin{aligned}(W_{C_1} \times l_{C_1}) + (W_{C_3} \times l_{C_3}) &= (2 \times 350) + (1 \times 350) \\ &= (700 + 350) \\ &= 1050 \ \textit{bytes}\end{aligned}$$

in each scheduling round, whereas, in DRR scheduling they can transmit up to

$$\begin{aligned}&\left(\left\lfloor \frac{(Q_{C_1} + \Delta_{C_1}^{max})}{l_{C_1}} \right\rfloor \times l_{C_1}\right) + \left(\left\lfloor \frac{(Q_{C_3} + \Delta_{C_3}^{max})}{l_{C_3}} \right\rfloor \times l_{C_3}\right) \\ &= \left(\left\lfloor \frac{(Q_{C_1} + l_{C_1} - 1)}{l_{C_1}} \right\rfloor \times l_{C_1}\right) + \left(\left\lfloor \frac{(Q_{C_3} + l_{C_3} - 1)}{l_{C_3}} \right\rfloor \times l_{C_3}\right) \\ &= \left(\left\lfloor \frac{(770 + 350 - 1)}{350} \right\rfloor \times 350\right) + \left(\left\lfloor \frac{(377 + 350 - 1)}{350} \right\rfloor \times 350\right) \\ &= 1050 + 700 = 1750 \ \textit{bytes}\end{aligned}$$

Thus these classes can be served earlier in case of DRR scheduling and when they are no more active C_2 is served at the maximum capacity.

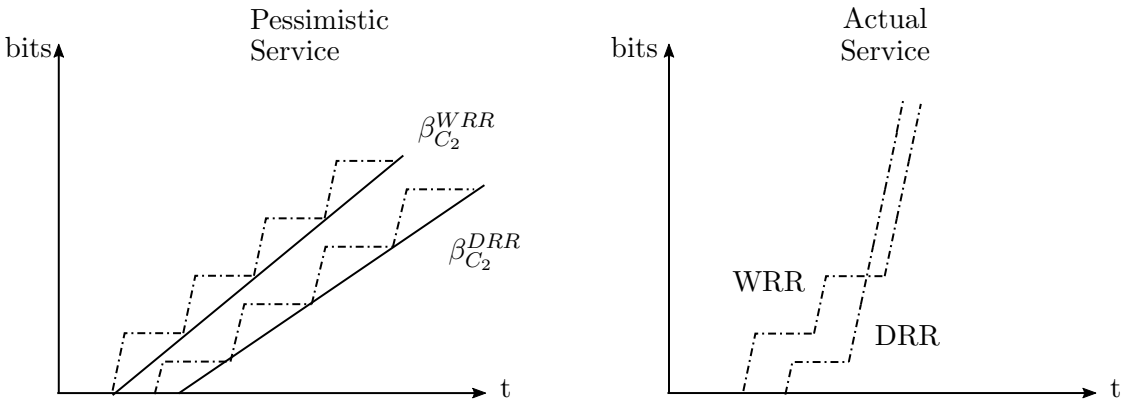


Figure 6.8 Illustration of C_2 flow service

Non-Critical class C_3: In case of C_3 flows the delay in WRR is less than that in DRR scheduling, thanks to the lower scheduler latency in WRR.

$$(\Theta_{C_3}^{WRR} = 88\ \mu sec) < (\Theta_{C_3}^{DRR} = 160.14\ \mu sec)$$
$$(\rho_{C_3}^{WRR} = 24.13\ bits/\mu sec) < (\rho_{C_3}^{DRR} = 24.38\ bits/\mu sec)$$

The scenario for C_3 flows is different from that discussed for C_1 and C_2 flows: since there is very less number of flows in C_3 it is possible that these flows are always delayed by C_1 and C_2 flows in each scheduling round which leads to a not so pessimistic service curve and the actual service received by these flows is similar to that considered by the service curve. Moreover, as shown in Figure 6.9, even if the service rate in DRR is slightly higher the active period of C_3 flows (represented by shaded area) can be too short to be benefited from it, whereas, in WRR these flows benefit the smaller scheduler latency.

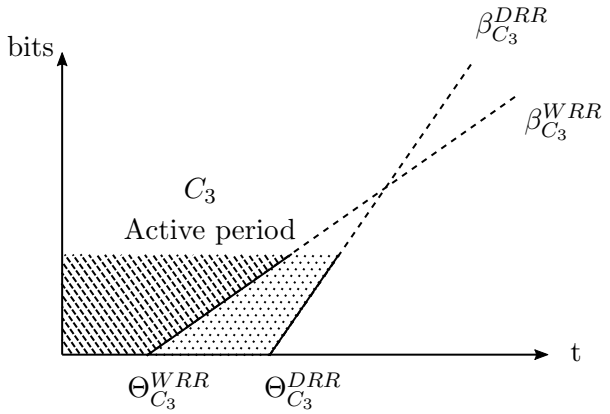


Figure 6.9 Illustration of C_3 flow service

6.4.2.2 WRR scheduling with heterogeneous flows

Let us now perform a similar analysis on the given network configuration but this time the flows are constrained to transmit frames of a certain size range. The differentiation of flows into two critical classes C_1 and C_2 is the same as the one discussed earlier, i.e. based on their BAG values (Table 6.4). The number of additional flows (C_3) is limited to 10 flows which are identical to C_1 flows. In the following paragraphs we will see how the difference between the worst-case delays in WRR scheduling and DRR scheduling increases when the range of frame lengths is increased in each class.

We analyse 4 different scenarios: initially the flows in critical classes (C_1 and C_2) are

constrained to transmit frames of unique size $l_{C_1}^{max} = l_{C_1}^{min} = 350$ bytes and $l_{C_2}^{max} = l_{C_2}^{min} = 400$ bytes, then in the second case C_1 flows can transmit frames of lengths in range $l_{C_1}^{max} = 350$ bytes and $l_{C_1}^{min} = 300$ bytes whereas C_2 flows can transmit frames of lengths in range $l_{C_2}^{max} = 400$ bytes and $l_{C_2}^{min} = 300$ bytes, then in the third case this range is further increased to $l_{C_1}^{max} = 400$ bytes, $l_{C_1}^{min} = 300$ bytes, $l_{C_2}^{max} = 500$ bytes and $l_{C_2}^{min} = 300$ bytes and so on (see table 6.5).

Table 6.5 Range of frame lengths per class

Case	Frame length range (bytes)		
	C_1	C_2	C_3
1	350	400	350
2	300 – 350	300 – 400	300 – 350
3	300 – 400	300 – 500	300 – 400
4	200 – 400	200 – 500	200 – 400

Since the delay experienced by different flows in each class is affected by the range of frame lengths transmitted in the network, the optimal quantum distribution in DRR scheduling that allows critical classes to meet their deadlines is also affected. So we compute the quantum distribution using Algorithm 2 separately for each case. This quantum distribution Q_{C_x} and the corresponding bandwidth $\rho_{C_x}^{DRR}$ available to each class are shown in Table 6.6. In WRR scheduling, the closest approximation of bandwidth allocation is computed based on the traditional weight assignment method discussed earlier. The weights W_{C_x} and bandwidth $\rho_{C_x}^{WRR}$ distribution are also shown in Table 6.6.

Let us observe the worst-case end-to-end delays computed in each case, which are also illustrated by Figure 6.10. Case 1 belongs to the scenario of homogeneous frame lengths and it is exactly same as the one discussed in Section 6.4.2.1 where the delay in critical classes C_1 and C_2 is lower in DRR as compared to WRR. In case 2, 3, and 4 the delays in C_1 and C_2 remains lower in DRR as compared to WRR (for the same reasons as explained earlier through Figure 6.7 and 6.8) and the difference between the two varies based on the difference between the bandwidth allocated to these classes in each scheduler. In C_3, the delays are lower in WRR as compared to DRR in cases 1, 2 and 3, since the bandwidth allocated in each scheduler is approximately similar and C_3 flows takes advantage of low scheduler latency in WRR. In case 4, C_3 delay is exceptionally increased and it is higher in WRR as compared to DRR, which is obvious since the weights computed by traditional approach led to a large difference between minimum bandwidth allocated to C_3 in WRR and DRR.

Table 6.6 Quantum, weight and bandwidth distribution per class in different cases

Case	C_x	Q_{C_x} (bytes)	$\rho_{C_x}^{DRR}$ (bits/μsec)	W_{C_x} (bytes)	$\rho_{C_x}^{WRR}$ (bits/μsec)
1	C_1	770	49.77	2	48.27
	C_2	400	25.85	1	27.58
	C_3	377	24.38	1	24.13
2	C_1	845	45.4	2	35.29
	C_2	400	21.49	1	17.64
	C_3	616	33.1	2	35.29
3	C_1	1032	47.33	2	31.57
	C_2	500	22.93	1	15.78
	C_3	648	29.72	2	31.57
4	C_1	1060	41.29	3	31.57
	C_2	500	19.47	1	9.09
	C_3	1007	39.22	2	19.04

In all these cases the critical classes C_1 and C_2 meet their deadlines in both DRR and WRR schedulers and the delay in non-critical class C_3 is lower (except in case 4) in WRR which makes it a better option to be used when the flows are constrained to transmit frame lengths within a small range. However, due to the absence of an efficient bandwidth allocation mechanism in WRR scheduling the performance guarantee may not be provided (as in case 4).

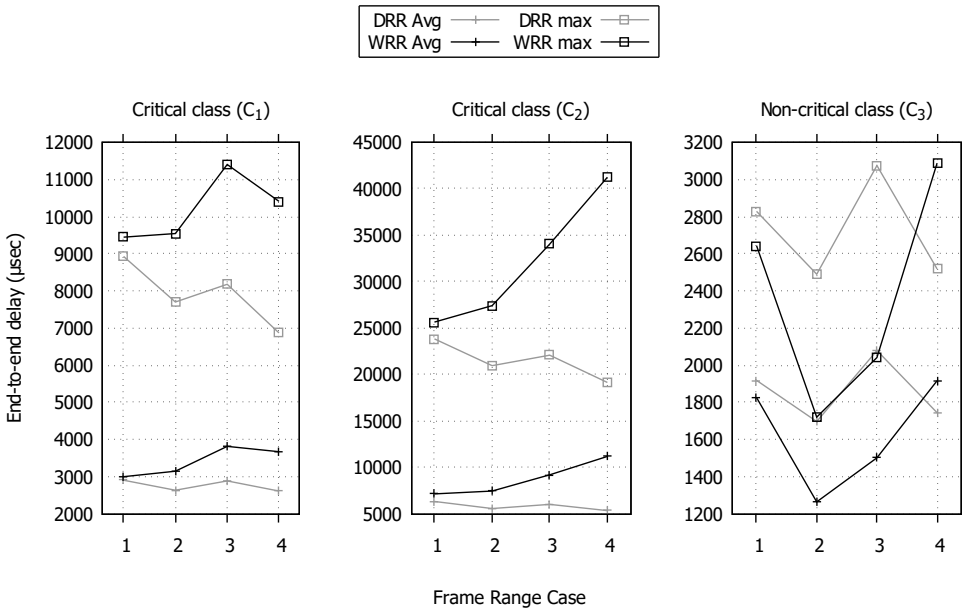


Figure 6.10 End-to-end delay comparison in DRR and WRR with heterogeneous flows.

6.5 Conclusion

In this chapter, we have extended our optimised NC approach for WCTT analysis of WRR scheduling based network. This approach is applied to compute the worst-case delay bounds in an industrial configuration of AFDX network. We have shown that in some cases it can be very difficult to perform the worst-case delay analysis when the WRR scheduling is used at the switch output ports.

We have also presented a comparison of the worst-case delay bounds obtained in the given AFDX network under WRR and DRR scheduling. The main difficulty in such comparison is the absence of a mechanism to achieve the desired bandwidth distribution in WRR scheduling.

Conclusion

In industrial real-time embedded systems, the real-time switched Ethernet network is a promising candidate to satisfy the increasing needs of communication. For instance, AFDX is dedicated solution based on switched Ethernet adapted in the context of aeronautics. In this network, the deterministic communication guarantee is provided thanks to Worst Case Traversal Time (WCTT) analysis using Network Calculus (NC). The pessimism involved by NC approach leads to over-dimensioning of the network architecture, which is lightly loaded. The manufacturers envision to use this available network capacity by allowing traffic from different functions of mixed-criticality.

In the presence of mixed criticality flows, the introduction of Quality-of-Service (QoS) mechanism is necessary to satisfy the constraints on critical flows. In this thesis, we have studied the worst-case delay analysis suitable for such QoS-aware real-time switched Ethernet network. We have specifically focused on DRR and WRR scheduling policies as they are envisioned by the industry for future industrial networks.

Recapitulation of contributions and obtained results

In mixed-criticality flow environment, the flows of high criticality must be protected from lower criticality flows to ensure hard real-time constraints. The use of priorities efficiently helps to meet more strict latencies for the most critical flows than what would be feasible with FIFO queues alone. However, static priority scheduling does not provide fair service to the flows of different priorities. In such an architecture with fixed priorities, there is a risk of bandwidth starvation for the lowest priority flows. The starvation problem can be avoided with differentiation service policies, such as DRR or WRR scheduling. These scheduling policies allow to guarantee a certain minimum share of bandwidth to different classes of flows. For instance, in DRR each class is assigned specific quantum which represents the number of bytes that can be transmitted from the respective class, irrespective of the traffic in competing classes. Moreover, reserving a

fair share of bandwidth for different classes can improve the overall performance of the network.

To achieve the goal of improving network bandwidth utilisation with DRR scheduling, we show in Chapter 3 how the bandwidth can be efficiently shared between critical and non-critical flow classes. We have proposed an algorithm that assigns the minimum quanta to critical classes that ensure that no deadlines will be missed, and eventually, it maximises the percentage of bandwidth assigned to non-critical flows. This algorithm relies on the classical NC approach for the computation of delay bounds in each flow class.

In chapter 4 we show that the classical NC approach can be pessimistic as it does not take into account the traffic from individual classes. It assumes maximum traffic in competing lasses which is limited by the quantum assigned to these classes in each service round. We have also proposed an optimised NC approach to mitigate this pessimism.

An evaluation on an industrial AFDX network configuration, given in Chapter 5, shows that our optimised NC approach gives a significant reduction in pessimism in worst-case delay bound computation as compared to classical NC approach, up to 40% for this configuration. We have also performed evaluation of the quantum assignment algorithm on the same network configuration. The results have shown that our algorithm successfully distributes quantum among flow classes that allows the guarantee on critical flows while minimising the delays for non-critical flows. The performed case study has also shown that the service provided by DRR scheduler is better than that by the SPQ scheduler in a mixed-criticality flow environment.

Another case study is performed in Chapter 6 to compare the worst-case delays obtained with DRR scheduling to that obtained with WRR scheduling on similar network configuration. The results show that in presence of flows with a big difference in frame sizes within the flow class the end-to-end delays in a DRR based network are smaller as compared to a WRR based network, but, the results in WRR based network are improved in case of higher homogeneity in frame sizes within each class.

Perspective

By the end of this thesis, numerous research perspectives are open. Some of them concern the remaining pessimism in NC approach. Since this pessimism leads to an over-dimensioning of the network architecture, it would be interesting to quantify this pessimism. Such a work has been done in [BSF10], where the pessimism of NC approach and Trajectories approach is upper bounded in the context of industrial AFDX config-

urations, thanks to the simulation of unfavourable scenarios. Another work given in [SLSF17] proposes a more generic approach to evaluate pessimism based on optimistic assumptions in the NC approach. However, these two works are limited to the switched Ethernet network with FIFO scheduling.

The quantum allocation algorithm proposed in this thesis depends on the classical NC approach for WCTT analysis, which is found to be very pessimistic. Which means the output of the algorithm is also affected by this pessimism. Therefore, we are currently working on the adaptation of this algorithm to the optimised NC approach.

Another future work is to develop a similar assignment algorithm for other policy like WRR scheduling.

Glossary

C_x	Traffic class.
$Q_{C_x}^h$	Quantum (number of bytes) of C_x at h.
R	Ethernet link rate.
T_i	Minimum inter-frame interval.
$W_{C_x}^h$	Weight (number of frames) of C_x at h.
$X_{C_x}^h$	Initial scheduler latency in DRR.
$Y_{C_x}^h$	Complementary scheduler latency in DRR.
$\Delta_{C_x}^h$	Deficit in C_x at h.
$\Theta_{C_x}^h$	Scheduler latency.
$\rho_{C_x}^h$	Theoretical service rate of C_x at h.
l_i^{max}	Maximum frame length in v_i.
l_i^{min}	Minimum frame length in v_i.
sl	Switching latency.

Acronyms

AFDX	Avionics Full-Duplex.
ARINC	Aeronautical Radio Incorporated.
AVB	Ethernet Audio/Video Bridge.
BAG	Bandwidth Allocation Gap.
CAN	Controller Area Network.
CBS	Credit Based Shaper.
COTS	Commercial Off-The-Shelf.
CSMA/CD	Carrier Sense Multiple Access with Collision Detection.
DRR	Deficit Round Robin.
EPL	Ethernet Powerlink.
FA	Forward Analysis.
FIFO	First-In First-Out.
MC	Model Checking.
NC	Network Calculus.
RR	Round Robin.
SP/FIFO	Strict Priority/First-In First-Out.
TA	Trajectory approach.
TCnet	Time-Critical Control Network.
TSN	Time-Sensitive Network.
TTEthernet	Time-Triggered Ethernet.
VLAN	Virtual Local Area Network.
VL	Virtual Link.
WCTT	Worst Case Traversal Time.

WRR Weighted Round Robin.

Bibliography

[AD94] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2), 1994.

[Air05] Airbus. Aircraft data network, parts 1,2,7 aeronotical radio inc. Technical report, ARINC Specification 664, 2002 - 2005.

[ASEF12] Muhammad Adnan, Jean-Luc Scharbarg, Jérôme Ermont, and Christian Fraboul. An improved timed automata approach for computing exact worst-case delays of afdx sporadic flows. In *17th International Conference on Emerging Technologies Factory Automation (ETFA)*, 09 2012.

[ASF11] Muhammad Adnan, Jean-Luc Scharbarg, and Christian Fraboul. Minimizing the search space for computing exact worst-case delays of afdx periodic flows. In *6th IEEE International Symposium on Industrial and Embedded Systems*, 06 2011.

[Bau11] Henri Bauer. *Analyse pire cas de flux hétérogènes dans un réseau embarqué avion.* Thesis Manuscript, October 2011.

[BBF+10] B. Berard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, and P. Schnoebelen. *Systems and Software Verification: Model-Checking Techniques and Tools.* Springer Publishing Company, Incorporated, 1st edition, 2010.

[Bel11] Lucia Lo Bello. The case for ethernet in automotive communications. *ACM SIGBED Rev.*, 2011.

[Bel14] Lucia Lo Bello. Novel trends in automotive networks: A perspective on ethernet and the ieee audio video bridging. *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, September 2014.

[Bou06] Jean-Yves Le Boudec. Application of network calculus to guaranteed service networks. *IEEE Trans. Inf. Theor.*, September 2006.

[BRBR17] Nassima Benammar, Frederic Ridouard, Henri Bauer, and Pascal Richard. Forward end-to-end delay analysis extension for fp/fifo policy in afdx networks. *22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, September 2017.

[BRBR18a] Nassima Benammar, Frederic Ridouard, Henri Bauer, and Pascal Richard. Forward end-to-end delay for afdx networks. *IEEE Transactions on Industrial Informatics*, 14(3):858–865, March 2018.

[BRBR18b] Nassima Benammar, Frederic Ridouard, Henri Bauer, and Pascal Richard. Timing analysis of avb ethernet network using the forward end-to-end delay analysis. *Proceedings of the 26th International Conference on Real-Time Networks and Systems*, 2018.

[BRKW17] Stefan Brunner, Jurgen Roder, Markus Kucera, and Thomas Waas. Automotive e/e-architecture enhancements by usage of ethernet tsn. *13th Workshop on Intelligent Solutions in Embedded Systems (WISES)*, June 2017.

[BSF10] Henri Bauer, Jean-Luc Scharbarg, and Christian Fraboul. Improving the worst-case delay analysis of an afdx network using an optimized trajectory approach. *IEEE Trans. Industrial Informatics*, 6(4), Nov 2010.

[BSS12] Marc Boyer, Giovanni Stea, and William Mangoua Sofack. Deficit round robin with network calculus. *Performance Evaluation Methodologies and Tools (VALUETOOLS), 2012 6th International Conference on (pp. 138-147). IEEE*, page 10, October 2012.

[BT12] Jean-Yves Le Boudec and Patrick Thiran. *Network Calculus: a theory of deterministic queuing systems for the internet*, volume 2050. LNCS, April 2012.

[CEL05] Vicent Cholvi, Juan Echagüe, and J.-Y LeBoudec. On the feasible scenarios at the output of a fifo server. *Communications Letters, IEEE*, 06 2005.

[Cru91] Rene L Cruz. A calculus for network delay. ii. network analysis. *IEEE Trans. Inf. Theor.*, 37(1), 1991.

[CSEF06] Hussein Charara, Jean-Luc Scharbarg, Jerome Ermont, and Christian Fraboul. Methods for bounding end-to-end delays on an afdx network. In *Proceedings of the 18th Euromicro Conference on Real-Time Systems*, ECRTS '06, Washington, DC, USA, 2006. IEEE Computer Society.

[Dec05] Jean-Dominique Decotignie. Ethernet-based real-time and industrial communications. *Proceedings of the IEEE*, June 2005.

[DH03] O Dolejs and Zdeněk Hanzálek. Simulation of ethernet for real-time applications. In *IEEE International Conference on Industrial Technology*, January 2003.

[Eth] Real-Time Ethernet. Tcnet (time-critical control network): Proposal for a publicly available specification for real-time ethernet.

[Eth04] Real-Time Ethernet. Ethernet control automation technology (ethercat): Proposal for a publicly available specification for real-time ethernet, 2004.

[Fel04] Joachim Feld. Profinet-scalable factory communication for all applications. In *IEEE International Workshop on Factory Communication Systems, 2004. Proceedings.*, pages 33–38. IEEE, 2004.

[FFG06] Fabrice Frances, Christian Fraboul, and Jérôme Grieu. Using network calculus to optimize the afdx network. In *ERTS 2006 : 3rd European Congress ERTS Embedded real-time software*, Toulouse, France, 2006.

[FJJ09] Xing Fan, Magnus Jonsson, and Jan Jonsson. Guaranteed real-time communication in packet-switched networks with fcfs queuing. *Computer Networks*, 02 2009.

[GKT14] Andy Gothard, Rick Kreifeldt, and Craig Turner. Avb for automotive use. *AVnu Alliance White Paper*, 2014.

[GRD02] Jean-Philippe Georges, Eric Rondeau, and Thierry Divoux. Evaluation of switched ethernet in an industrial context by using the network calculus. In *4th IEEE International Workshop on Factory Communication Systems*, 02 2002.

[Gri04] Jérôme Grieu. *Analyse et évaluation de techniques de commutation Ethernet pour l'interconnexion des systèmes avioniques.* Thesis Manuscript, September 2004.

[Gro] Ethernet POWERLINK Standardization Group. Ethernet powerlink. `https://www.ethernet-powerlink.org/downloads/technical-documents`. Accessed: 24-06-2019.

[HL17] Feng He and Ershuai Li. Deterministic bound for avionics switched networks according to networking features using network calculus. *Chinese Journal of Aeronautics*, 2017.

[HMVdK13] Peter Hank, Steffen Muller, Ovidiu Vermesan, and Jeroen Van den Keybus. Automotive ethernet: In-vehicle networking and smart mobility. In *Design, Automation Test in Europe Conference Exhibition (DATE)*, 01 2013.

[JNTW04] Juergen Jasperneite, Peter Neumann, Michael Theis, and Kym Watson. Deterministic real-time communication with switched ethernet. *4th IEEE International Workshop on Factory Communication Systems*, 09 2004.

[KAGS05] Hermann Kopetz, Astrit Ademaj, Petr Grillinger, and Klaus Steinhammer. The time-triggered ethernet (tte) design. In *8th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing (ISORC'05)*, pages 22–33. IEEE, 2005.

[KRBR15] Georges Kemayo, Frederic Ridouard, Henri Bauer, and Pascal Richard. Improving afdx end-to-end delays analysis. *20th IEEE Conference on Emerging Technologies Factory Automation (ETFA)*, 2015.

[KS02] Salil S. Kanhere and Harish Sethu. On the latency bound of deficit round robin. In *Proceedings. Eleventh International Conference on Computer Communications and Networks*, Oct 2002.

[KZST11] Andreas Kern, Hongyan Zhang, Thilo Streichert, and Jürgen Teich. Testing switched ethernet networks in automotive embedded systems. In *6th IEEE International Symposium on Industrial and Embedded Systems*, 06 2011.

[LH04a] J Loeser and H Haertig. Low-latency hard real-time communication over switched ethernet. In *16th Euromicro Conference on Real-Time Systems*, 01 2004.

[LH04b] J Loeser and H Haertig. Using switched ethernet for hard real-time communication. In *Parallel Computing in Electrical Engineering, 2004. International Conference on*, 2004.

[LHWC12] Hyung-Taek Lim, Daniel Herrscher, Martin Waltl, and Firas Chaari. Performance analysis of the ieee 802.1 ethernet audio/video bridging standard. *5th International ICST Conference on Simulation Tools and Techniques*, 2012.

[Li13] Xiaoting Li. *Ph.D. Thesis: Worst-case delay analysis of real-time switched Ethernet networks with flow local synchronization.* Thesis Manuscript, September 2013.

[LMS02] Luciano Lenzini, Enzo Mingozzi, and Giovanni Stea. Aliquem: a novel drr implementation to achieve better latency and fairness at o(1) complexity. *IEEE 2002 Tenth IEEE International Workshop on Quality of Service (Cat. No.02EX564)*, 05 2002.

[LMS05] Luciano Lenzini, Enzo Mingozzi, and Giovanni Stea. Delay bounds for fifo aggregates: A case study. *Computer Communications*, 02 2005.

[Lov15] Andrew Loveless. Ttethernet for integrated spacecraft networks, 2015.

[MFF07] Ahlem Mifdaoui, Fabrice Frances, and Christian Fraboul. Real-time characteristics of switched ethernet for "1553b" -embedded applications : simulation and analysis. In *SIES 2007 : IEEE 2nd Second International Symposium on Industrial Embedded Systems*, Lisbon, Portugal, July 2007.

[MM05] Steven Martin and Pascale Minet. Holistic and trajectory approaches for distributed non-preemptive fp/dp* scheduling. *4th International Conference on Networking - Volume Part I*, 2005.

[Mod04] IDA Modbus. Modbus messaging on tcp/ip implementation guide. *v1. 0a, June*, 4, 2004.

[MVNB18] Jörn Migge, Josetxo Villanueva, Nicolas Navet, and Marc Boyer. Insights on the performance and configuration of avb and tsn in automotive ethernet networks. *Proceedings of Embedded Real-Time Software and Systems (ERTS 2018)*, 2018.

[PG93] Abhay K. Parekh and Robert G. Gallager. A generalized processor sharing approach to flow control in integrated services networks: the single-node case. *IEEE/ACM Transactions on Networking*, June 1993.

[Sch01] Viktor Schiffer. The cip family of fieldbus protocols and its newest member-ethernet/ip. In *ETFA 2001. 8th International Conference on Emerging Technologies and Factory Automation. Proceedings (Cat. No. 01TH8597)*, pages 377–384. IEEE, 2001.

[SF07] Jean-Luc Scharbarg and Christian Fraboul. Simulation for end-to-end delays distribution on a switched ethernet. In *IEEE Conference on Emerging Technologies and Factory Automation (EFTA)*, 10 2007.

[SKS11] Till Steinbach, Franz Korf, and Thomas Schmidt. Real-time ethernet for automotive applications: A solution for future in-car networks. In *Digest of Technical Papers - IEEE International Conference on Consumer Electronics*, 09 2011.

[SLSF17] Aakash Soni, Xiaoting Li, Jean-Luc Scharbarg, and Christian Fraboul. Work in progress paper: pessimism analysis of network calculus approach on afdx networks. *12th IEEE International Symposium on Industrial Embedded Systems (SIES)*, June 2017.

[SLSF18a] Aakash Soni, Xiaoting Li, Jean-Luc Scharbarg, and Christian Fraboul. Integrating offset in worst case delay analysis of switched ethernet network with deficit round robbin. In *23rd International Conference on Emerging Technologies and Factory Automation (ETFA)*, pages 353–359, Sep. 2018.

[SLSF18b] Aakash Soni, Xiaoting Li, Jean-Luc Scharbarg, and Christian Fraboul. Optimizing network calculus for switched ethernet network with deficit round robin. In *2018 IEEE Real-Time Systems Symposium (RTSS)*, pages 300–311, Dec 2018.

[SLSF18c] Aakash Soni, Xiaoting Li, Jean-Luc Scharbarg, and Christian Fraboul. Wctt analysis of avionics switched ethernet network with wrr scheduling. In *Proceedings of the 26th International Conference on Real-Time Networks and Systems*, pages 213–222, 2018.

[Spe91] CAN Specification. Bosch. *Robert Bosch GmbH, Postfach*, 50, 1991.

[SRF09] Jean-Luc Scharbarg, Frédéric Ridouard, and Christian Fraboul. A probabilistic analysis of end-to-end delays on an afdx avionic network. *Industrial Informatics, IEEE Transactions on*, 5, Mars 2009.

[SSE19] Aakash Soni, Jean-Luc Scharbarg, and Jérôme Ermont. Quantum assignment for qos-aware afdx network with deficit round robin. In *Proceedings of the 27th International Conference on Real-Time Networks and Systems*, page 70–79. Association for Computing Machinery, 2019.

[SV96] Madhavapeddi Shreedhar and George Varghese. Efficient fair queuing using deficit round-robin. *IEEE/ACM Transactions on networking, 1996, vol. 4, no 3, p. 375-385*, page 11, 1996.

[VS96] Anujan Varma and Dimitrios Stiliadis. Latency-rate servers: a general model for analysis of traffic scheduling algorithms. *IEEE/ACM Transactions on Networking*, 03 1996.

List of figures

1.1 An example of a switched Ethernet network . . . 13
1.2 Characteristics of sporadic and periodic flows . . . 15
1.3 End-to-end delay illustration . . . 17
1.4 Possible frame sequences . . . 18
1.5 End-to-end delay results for a flow by different approaches . . . 21
1.6 NC curves and bounds . . . 24
1.7 Jitter integration in arrival curve . . . 28
1.8 Illustration of serialisation effect in v_1 and v_2 at S_2 . . . 30
1.9 Effect of additional traffic in FIFO scheduling . . . 33
1.10 A switch output port with SPQ scheduler . . . 34
1.11 Credit based shaping of high priority flow . . . 35
1.12 A switch output port with Round Robin scheduler . . . 36
1.13 WRR scheduling . . . 37
1.14 DRR scheduling . . . 39

2.1 An example of a switched Ethernet network with DRR scheduling . . . 44
2.2 Illustration of DRR scheduling . . . 47
2.3 Theoretical and actual service rate in C_2 . . . 49
2.4 DRR scheduler latency . . . 51
2.5 Illustration of DRR scheduler latency at S_1 output port . . . 52
2.6 Illustration of the impact of reduced service in C_1 . . . 54
2.7 Worst-case service for C_1 in DRR scheduler . . . 57

3.1 Network architecture . . . 60
3.2 A switched Ethernet network example . . . 61
3.3 Delay computation for flow v_2 of C_1 . . . 64
3.4 Service curves . . . 73

4.1 A switched Ethernet network with DRR scheduling 83
4.2 DRR scheduling rounds at output port (Figure 4.1) 84
4.3 Pessimism in NC service curve 86
4.4 Pessimistic Service Curve of C_1 at S_2 (Figure 4.1) 88
4.5 Lower bound on maximum service load in DRR scheduler 89
4.6 Effective load of C_2 and C_3 flows 91
4.7 A recall of switched Ethernet network in Figure 4.1 93
4.8 Frame sequences at e_1 with and without temporal separation of v_{11} and v_{13} 94
4.9 v_{11} and v_{13} frame sequence at S_1 94
4.10 Relative offset between v_{11} and v_{13} 96
4.11 Overall arrival curve at e_1 in classical NC approach 97
4.12 Aggregated arrival curves at e_1 98
4.13 Aggregated arrival curves at S_1 100
4.14 Overall arrival curve of C_3 flows at S_1 and S_2 100

5.1 AFDX network architecture 106
5.2 End-to-end delay bounds in C_1 and C_2 flows under FIFO scheduling . . . 109
5.3 End-to-end delay comparison in C_3 flows under FIFO and SPQ 113
5.4 End-to-end delay comparison in C_3 flows under FIFO, SPQ and DRR . . 114
5.5 Comparison of end-to-end delay bounds computed by classical NC and optimised NC approach 115

6.1 An example switch with WRR scheduling at output port 122
6.2 WRR scheduling rounds at output port (Figure 6.1) 123
6.3 NC service curve for C_1 with WRR scheduling 130
6.4 C_1 service with reduce traffic from C_2 and C_3 131
6.5 Lower bound on maximum service load in WRR scheduler 133
6.6 End-to-end delay comparison in DRR and WRR with homogeneous flows. 141
6.7 Illustration of C_1 flow service 142
6.8 Illustration of C_2 flow service 143
6.9 Illustration of C_3 flow service 144
6.10 End-to-end delay comparison in DRR and WRR with heterogeneous flows. 147

List of tables

2.1 Flow specifications 45

3.1 Flow and class specifications 62
3.2 Delay variation with quanta 66
3.3 Quantum assignment in the network (Figure 3.2). 77

4.1 Flow specifications 83
4.2 Definite offset for network (Figure 4.7) 97
4.3 End-to-end delays comparison 102

5.1 Flow characteristics (a) BAG (b) Frame length 106
5.2 Length of VL paths 107
5.3 VL parameters in industrial configuration 107
5.4 VL parameters in best-effort class C_3 109
5.5 End-to-end delay bounds in C_1 and C_2 flows under FIFO scheduling in presence of additional non-critical (C_3) class flows 110
5.6 End-to-end delay bounds in C_1 and C_2 flows under SPQ scheduling in presence of additional non-critical (C_3) class flows 112

6.1 Flow specifications 121
6.2 Reacall of flow characteristics in industrial configuration (a) BAG (b) Frame length 136
6.3 Flow differentiation in industrial configuration 136
6.4 Flow differentiation in industrial configuration with homogeneous flows . . 140
6.5 Range of frame lengths per class 145
6.6 Quantum, weight and bandwidth distribution per class in different cases . 146

Abstract / Résumé

EN: AFDX serves as a backbone network for transmission of critical avionic flows. This network is certified thanks to the WCTT analysis using Network Calculus (NC) approach. However, the pessimism introduced by NC approach often leads to an over-sized and eventually an under-utilized network. The manufacturers envision to better use the available network resources by increasing occupancy rate of the AFDX network by allowing additional traffic from other critical and non-critical functions. Such harmonization of AFDX network with mixed criticality flows necessitates the use of QoS mechanism to satisfy the delay constraints in different classes of flow. In this thesis we study such QoS-aware network, in particular, based on DRR and WRR scheduling. We propose an optimal bandwidth distribution method that ensures the service required by critical flows while providing maximum service to other non-critical flows. We also propose an optimized NC approach to compute tight delay bounds. Our approach has led to computation of up to 40% tighter bounds, in an industrial AFDX configuration, as compared to the classical approach.

FR: L'AFDX est utilisé comme un réseau fédérateur pour la transmission des flux avioniques critiques. Ce réseau est certifié grâce à l'analyse pire-cas utilisant l'approche Network Calculus (NC). Le pessimisme introduit par NC conduit souvent à un réseau surdimensionné et éventuellement sous-utilisé. Les avionneurs envisagent d'augmenter l'utilisation des ressources du réseau AFDX en ajoutant du trafic supplémentaire provenant d'autres fonctions critiques et non critiques. Le partage du réseau AFDX avec des flux de criticité mixtes nécessite l'utilisation d'un mécanisme de qualité de service (QoS) pour satisfaire les contraintes de délai des différentes classes de flux. Dans cette thèse, nous étudions un tel réseau déployant de la qualité de service, en particulier, basé sur l'ordonnancement DRR et WRR. Nous proposons une méthode optimale de distribution de la bande passante qui assure le service requis par les flux critiques tout en fournissant un service maximisé aux flux non-critiques. Nous proposons également une approche NC optimisée qui, sur une configuration industrielle de réseau AFDX, a permis de réduire les bornes jusqu'à 40%.

Keywords / Mots clés : AFDX, Quality of Service (QoS), Embedded Network, Deficit Round Robin (DRR), Weighted Round Robin (WRR), WCTT analysis, Network Calculus