Université Fédérale
Toulouse Midi-Pyrénées

THÈSE

En vue de l'obtention du

DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par l'Institut Supérieur de l'Aéronautique et de l'Espace

Présentée et soutenue par

Ludovic THOMAS

Le 12 septembre 2022

Analyse des conséquences sur les bornes de latences des combinaisons de mécanismes d'ordonnancement, de redondance et de synchronisation dans les réseaux temps-réel.

Ecole doctorale : **EDMITT - Ecole Doctorale Mathématiques, Informatique et Télécommunications de Toulouse**

Spécialité : **Informatique et Télécommunications**

Unité de recherche :
ISAE-ONERA MOIS MOdélisation et Ingénierie des Systèmes

Thèse dirigée par
Ahlem MIFDAOUI et Jean-Yves LE BOUDEC

Analysis of the side-effects on latency bounds of combinations of scheduling, redundancy and synchronization mechanisms in time-sensitive networks.

Abstract — Time-sensitive networks, as in the context of IEEE Time-Sensitive Networking (TSN) and IETF Deterministic Networking (DetNet), support safety-critical applications by providing deterministic services with guaranteed latency bounds. Several mechanisms, such as schedulers and traffic regulators (TSN ATS, asynchronous traffic shaping), were developed, and their effects on worst-case latency bounds have been widely studied in the literature by using the network-calculus framework. Time-sensitive networks are also required, however, to offer an easy reconfiguration with alternative paths, a high level of reliability and sometimes a time-synchronization service. To meet these needs, multi-path topologies were developed to enable the reconfiguration, and a set of redundancy and synchronization mechanisms were developed to provide high reliability and time synchronization. Although the mechanisms can rely on their own theory to validate their adequacy to their respective objective, their side effects on the latency bounds and their interactions with the schedulers and traffic regulators have hardly been analyzed in the literature. In this thesis, we use the network-calculus framework to analyze the combinations of mechanisms and their effects on latency bounds in time-sensitive networks with multi-path topologies.

Our main contributions at the theoretical level are as follows: 1/ We develop an algorithm (FP-TFA) for computing latency bounds in networks with multi-path topologies that lead to cyclic dependencies. We propose and analyze the partial-deployment approach of traffic regulators (either per-flow or TSN ATS) and another algorithm (LCAN) for breaking all cyclic dependencies at minimal cost. 2/ We analyze the effect of redundancy mechanisms on latency bounds by capturing their behavior with the network-calculus framework. We analyze their interactions with traffic regulators. We find that TSN ATS can yield unbounded latencies when used with redundancy mechanisms. 3/ We provide a time model that captures the clock non-idealities of synchronized and non-synchronized networks within the network-calculus framework. We show that traffic regulators with non-ideal clocks can lead to latency penalties; with TSN ATS this penalty is unbounded even in tightly synchronized networks. We propose two methods (rate-and-burst cascade and ADAM) for adapting the parameters of the traffic regulators and for addressing the above issue.

We also provide contributions of practical interest: a) the xTFA modular tool that computes latency bounds by using the results of the thesis, b) a module for simulating the effects of local clocks in the discrete-event simulator ns-3, and c) an application of our results to an industrial use-case.

Keywords:

time-sensitive networks; network calculus; traffic regulators; redundancy mechanisms; non-ideal clocks; time synchronization

Analyse des conséquences sur les bornes de latences des combinaisons de mécanismes d'ordonnancement, de redondance et de synchronisation dans les réseaux temps-réel.

Résumé — Les réseaux temps-réels, comme ceux spécifiés par IEEE *Time-Sensitive Networking* (TSN) et IETF *Deterministic Networking* (DetNet), fournissent aux applications critiques un service déterministe avec des bornes de latence garanties. Plusieurs mécanismes comme les ordonnanceurs et les régulateurs de trafic (TSN ATS, *asynchronous traffic shaping*) ont été développés et leurs effets sur les bornes de latences pire-cas ont été abondamment étudiés dans la littérature en utilisant la théorie du calcul réseau. Toutefois, les réseaux temps-réels doivent désormais aussi offrir une reconfiguration simplifiée avec des chemins alternatifs, un haut niveau de fiabilité et parfois un service de synchronisation du temps. Pour répondre à ces besoins, l'utilisation de topologies à plusieurs chemins a été encouragée pour faciliter la reconfiguration et des mécanismes de redondance et de synchronisation ont été développés pour fournir un haut niveau de fiabilité et une synchronisation du temps. Tandis que chacun de ces mécanismes dispose d'une théorie pour valider son efficacité dans son objectif respectif, la littérature n'a que peu étudié leurs effets secondaires sur les bornes de latences et leurs interactions avec les ordonnanceurs et les régulateurs de trafic. Dans cette thèse, nous utilisons la théorie du calcul réseau pour analyser les combinaisons de mécanismes et leurs effets sur les bornes de latences dans les réseaux temps-réel avec des topologies à plusieurs chemins. Nos principales contributions sur le plan théorique sont : 1/ Nous développons un algorithme (FP-TFA) qui calcule des bornes de latence dans les réseaux dans lesquels la variété des chemins crée des dépendances cycliques. Nous proposons et analysons l'approche de déploiement partielle des régulateurs de trafic (soit par flux, soit avec TSN ATS) ainsi qu'un autre algorithme (LCAN) qui casse toutes les dépendances cycliques à coût minimal. 2/ Nous analysons les effets des mécanismes de redondance sur les bornes de latence en modélisant leur comportement dans la théorie du calcul réseau. Nous analysons aussi leurs interactions avec les régulateurs de trafic. En particulier, nous observons que TSN ATS peut mener à des latences non bornées lorsqu'il est utilisé avec les mécanismes de redondances. 3/ Nous proposons un modèle d'horloge qui décrit, au sein de la théorie du calcul réseau, les imperfections des horloges des réseaux synchronisés ou non. Nous montrons que l'usage de régulateurs de trafic avec des horloges imparfaites occasionne une pénalité dans les bornes de latence. Avec TSN ATS, cette pénalité n'est pas bornée, y compris dans les réseaux synchronisés avec une grande précision. Nous proposons deux méthodes (cascade et ADAM) pour adapter les paramètres des régulateurs et ainsi résoudre ce problème.

Nous fournissons également des contributions d'intérêt pratique : a) l'outil modulaire xTFA, qui calcule des bornes de latences en utilisant les résultats de la thèse, b) un module pour simuler l'effet des horloges locales dans le simulateur à évènements discrets ns-3, et c) une application de nos résultats sur une étude de cas industrielle.

Mots clés :

réseaux temps-réel; calcul réseau; régulateurs de trafic; mécanismes de redondance; horloges imparfaites; synchronisation

Remerciements

Voilà donc les dernières lignes que l'on écrit, les dernières touches avant de laisser la thèse pour de bon derrière soi. Mais ce résultat final n'est en rien l'œuvre d'une seule personne. C'est le résultat de plusieurs années de travail avec des personnes extraordinaires. À toutes ces personnes, merci.

Je tiens tout d'abord à remercier celles et ceux qui ont accepté de prendre du temps dans leurs plannings chargés pour évaluer mon travail et dont les questions et remarques ont permis les touches finales apportées à ce manuscrit. Merci au Prof. Ye-Qiong Song d'avoir accepté d'être président de mon jury. Je le remercie ainsi que le Prof. Jens B. Schmitt pour avoir accepté de relire mon manuscrit et pour leurs précieux retours. Je remercie également Dre. Liliana Cucu-Grosjean et Dre. Claire Pagetti pour avoir accepté de faire partie de mon jury, pour leurs précieux retours et pour nos discussions qui ont suivi.

J'adresse un merci tout particulier à ma directrice de thèse et à mon directeur de thèse, sans qui cette page de remerciements n'existerait pas. Merci à la Prof. Ahlem Mifdaoui pour avoir remué ciel et terre - de la recherche de financement à la constitution du jury - afin que je puisse réaliser cette thèse. Merci pour nos nombreuses discussions, pour m'avoir inclus dans des projets industriels, merci pour tout ce temps passé à co-écrire les articles et pour tous ses conseils. Merci au Prof. Jean-Yves Le Boudec pour m'avoir accueilli et ré-accueilli au sein du LCA2. Chaque jour - depuis celui où il répondait positivement à un e-mail de la Prof. Mifdaoui il y a déjà quatre ans, en passant par ce jour férié où nous devions être quasiment les deux seuls sur le campus pour finir un papier - je mesure le privilège et la chance que j'ai de travailler au sein de son laboratoire. Je les remercie enfin tous deux pour leur confiance en moi.

Au delà de ces dernières années de thèse, ce manuscrit est aussi le résultat d'un parcours marqué par de nombreuses rencontres avec des personnes formidables à la passion contagieuse. Merci ainsi à M. Brice Dellandréa et à son équipe au sein de Thales Alenia Space, au Prof. Emmanuel Lochin, au Dr. Nicolas Kuhn et à toute son ancienne équipe au CNES. Merci à tous mes collègues de Toulouse à Lausanne, merci notamment à tous les anciens doctorants de l'équipe ResCom de l'ISAE et merci à mes amis de Toulouse. D'une manière générale, merci à toutes les personnes formidables que j'ai eut le plaisir de rencontrer et avec qui j'ai eut le plaisir de refaire le monde autour d'un café.

Enfin, comment ne pas dédier cette thèse à mes proches qui ont gardé confiance et patience ces jours où j'avais perdu l'une et l'autre. Chacune de ces pages est imprégnée de leur soutien constant, de leur bienveillance. Je pense ainsi à mes parents à qui j'adresse un infini merci pour tant de choses qui ne peuvent être contenues en quelques lignes. Je pense enfin à Alexis: merci d'avoir franchi les tempêtes avec moi, merci pour ton soutien, merci d'être là.

"*Certains [sentiments] sont tellement forts que peu importe les lettres que l'on assemblera, cela ne suffira pas à dire ce que l'on ressent.*"

Victor Hugo

Contents

I Context **3**

1 Technological Context: Time-Sensitive Networks **5**
- 1.1 Service Performance of Communication Networks 6
- 1.2 Technological Trends in Time-Sensitive Networks 7
- 1.3 The IEEE TSN Task Group 10
- 1.4 The IETF DetNet Working Group 12
- 1.5 Performance Analysis of Time-Sensitive Networks 12

2 Theoretical Context: Network Calculus **17**
- 2.1 The Underlying Formalism: Min-Plus Algebra 18
- 2.2 Main Concepts of Network Calculus 20
- 2.3 The Main Results of the Network-Calculus Theory 22
- 2.4 FIFO-per-Class Systems 24
- 2.5 Compositional Approaches for the Analysis of Feed-Forward FIFO-per-class Networks 26
- 2.6 Network-Calculus Model For Time-Sensitive Networks 30

II Theoretical Contributions **35**

3 Cyclic Dependencies **37**
- 3.1 Related Work 38
- 3.2 LCAN: an Optimal Algorithm for Breaking All Cyclic Dependencies 45
- 3.3 FP-TFA: Computing Tight Delay Bounds in Networks with Cyclic Dependencies 50
- 3.4 Evaluation of the Partial-Deployment Approach 58

4 Packet Replication and Elimination **63**
- 4.1 Related Work on Packet Replication And Elimination 64
- 4.2 Modeling the Redundancy Mechanisms 69
- 4.3 Toolbox for the Deterministic Analysis of Packet Replication and Elimination Functions 75
- 4.4 Interactions between PREF and Traffic Regulators 82
- 4.5 Evaluation of the Performance of The Toolbox 88

5 Effects of Clock Non-Idealities **93**
- 5.1 Related Work 95
- 5.2 Time Model 96
- 5.3 Network-Calculus Toolbox for Networks With Non-Ideal Clocks 100
- 5.4 Computing End-To-End TAI Bounds in Time-Sensitive Networks 104
- 5.5 Regulators in Networks With Non-Ideal Clocks 107
- 5.6 Evaluation of the Approaches 114

III Practical Contributions **117**

6 xTFA: A Tool for Computing Delay Bounds in Time-Sensitive Networks **119**
6.1 Overview of xTFA 119
6.2 Main Original Ideas in experimental modular TFA (xTFA) for Feed-Forward Networks 121
6.3 Adaptations of xTFA for Cyclic Dependencies 134

7 Implementation of Local Clocks in ns-3 **137**
7.1 ns-3, a Network Simulator 138
7.2 Previous Work, Limitations and New Requirements 142
7.3 Proposed Design for the Implementation of Clocks in ns-3 145
7.4 Application: Simulating the IR Instability with Non-Ideal Clocks 150

8 Application of the Contributions to an Industrial Use-Case **153**
8.1 Description of the Industrial Use-Case 153
8.2 Side-Effects of the Multi-Path Topology on Latency Bounds: Contributions of Chapter 3 154
8.3 Side-Effects of the Redundancy Mechanisms on Latency Bounds: Contributions of Chapters 4 and 6 156
8.4 Side-Effects of the Synchronization on Latency Bounds: Contributions of Chapters 5 and 6 157

IV Conclusion and Perspectives **161**

9 Conclusion and Perspectives **163**
9.1 Summary of the Contributions 163
9.2 Perspectives 166

V Résumé en Français du Contexte et des Contributions Principales **169**

VI Appendices **179**

A Vade Mecum **181**
A.1 List of Acronyms 181
A.2 Glossary 182
A.3 General Notations 185

B Proofs **189**
B.1 Proofs of Chapter 3 189
B.2 Proofs of Chapter 4 190
B.3 Proofs of Chapter 5 225

Bibliography **241**

List of Figures

1 Illustration of our theoretical contributions in the thesis 2

1.1 Evolution of the paradigms used in time-sensitive networks for sharing the network resources between the end systems 7
1.2 Classification of the past and future trends of time-sensitive networks in two industrial sectors 8
1.3 Illustration of the distribution of the latency in a network 13

2.1 Example of curves used in this thesis 19
2.2 A flow f crosses a system S 20
2.3 Graphical illustration of the arrival-curve constraint 21
2.4 Graphical illustration of the service-curve constraint 22
2.5 A flow f that crosses a network element S 22
2.6 A flow f crosses two network elements S_1, S_2 in sequence. 23
2.7 The "for-free" properties of the *greedy shaper* and of the *packetizer*. 23
2.8 Toy example for the introduction of the different compositional approaches. . 26
2.9 Illustration of the boundary ϕ of the network. 30
2.10 Model for any device of the time-sensitive network. 30
2.11 Overview of the forwarding process in an IEEE TSN output port. 32
2.12 Relation between the bridges, the vertices and the observation points. 33

3.1 Phenomenon of burstiness increase. 38
3.2 Toy example of a network with cyclic dependencies. 39
3.3 Shaping-for free property of two types of regulators. 42
3.4 Detailed model of the vertex $\mathrm{S1}_{\mathrm{East}}$ with a PFR. 43
3.5 Overview of the LCAN algorithm. 45
3.6 A PFR is placed for the toy example at $\mathrm{S0}_{\mathrm{East}}$ 46
3.7 An IR is placed for the toy example at $\mathrm{S0}_{\mathrm{East}}$. 47
3.8 Excerpt of the burst-dependency graph of the toy example. 47
3.9 The *burst-dependency graph* for the toy example. 49
3.10 Detailed model for the computation of the delay bound through vertex $\mathrm{S1}_{\mathrm{East}}$ 52
3.11 Contribution of the parent $\mathrm{S4}_{\mathrm{North}}$ in the aggregate arrival curve at the input of the CBQS within $\mathrm{S1}_{\mathrm{East}}$ 54
3.12 Virtual cut $(\mathrm{S4}_{\mathrm{North}}, \mathrm{S1}_{\mathrm{East}})$ 56
3.13 Overview of the FP-TFA algorithm when it searches for a fixed-point 56
3.14 Model for $\mathrm{S1}_{\mathrm{East}}$ when a regulator is placed for the parent $\mathrm{S4}_{\mathrm{North}}$ 57
3.15 Parametric topology used for the evaluation 58
3.16 Number of regulators versus the grid network size for the different approaches. 59
3.17 End-to-end latency bound of the flows as a function of the network load . . . 60

4.1 Physical topology used for the toy example throughout the chapter. 64
4.2 Path of a replicated flow f. 65

4.3 An excerpt of the flow graph $\mathcal{G}(f)$ for the toy example used throughout the chapter. 66
4.4 An example trajectory on the toy example of Figure 4.3 that causes the output traffic after the PEF to exhibit a double rate for some period of time. 68
4.5 The toy example of Figure 4.3, extended with a POF and a REG to deal with the mis-ordering and burstiness increase issues due to PEF. 69
4.6 The entire flow-graph $\mathcal{G}(f)$ of the flow f in the toy example 69
4.7 Illustration of different positions for the PEF in the toy example. 70
4.8 Functional model of the POF. 72
4.9 Model of a REG. 73
4.10 Example of an organization of the optional functions within an output port. . 74
4.11 Another example of the organization of the optional functions within an output port. 75
4.12 Arrival curve at the output of the PEF on the toy example, and tightness of the result. 77
4.13 Notations of Corollary 4.1. 77
4.14 Trajectory showing that the result of Corollary 4.1 is tight for the toy example. 78
4.15 Toy example of Figure 4.3, with a POF placed after the PEF 79
4.16 Trajectory of the packets at the output of the POF of Figure 4.15. 80
4.17 Notations for the analysis of the interactions between PEF and a PFR. . . . 82
4.18 Toy example of Figure 4.3 with a PFR placed after the PEF 83
4.19 An acceptable trajectory on the toy example, that shows that the delay bound D' through $\mathcal{S}'$ is at least 14 t.u. 83
4.20 Notations for the analysis of the interactions between PEF and an IR. 84
4.21 Toy example of Figure 4.3, to which we added a POF followed by a PFR. . . 87
4.22 Output of the POF and of the PFR of Figure 4.21 when they process the trajectory of Figure 4.19. 87
4.23 Notations of Theorem 4.6 88
4.24 Parametric network for the evaluation of the delay bounds obtained with the *intuitive approach* and with xTFA. 88
4.25 Comparison of the end-to-end delay bounds obtained on the parametric topology with each method. 89

5.1 Time function h_i between the TAI and the clock time. 97
5.2 Envelope of $d(t)$ for the non-synchronized and synchronized time models. . . 100
5.3 Clocks $\mathcal{H}_g$ and $\mathcal{H}_i$ observe flow f that enters system S. 101
5.4 Changing the observing clock for a leaky-bucket arrival curve. 102
5.5 Changing the observing clock for rate-latency and leaky-bucket service curves. 104
5.6 Flow graph for the flow of the toy example. 104
5.7 Illustration of the strategy "everything in $\mathcal{H}_{\text{TAI}}$" for the example of Figure 5.6. 105
5.8 Illustration of the strategy "always in local time" for the example of Figure 5.6. 106
5.9 Network model for a regulated flow. 108
5.10 An example of mapping between the flow graph and the sequence of S_k systems. 108
5.11 Notations of Proposition 5.10 113
5.12 Comparison of the end-to-end strategies and of the adaptation methods . . . 114

6.1 Overview of the TFA model for vertex $\mathrm{S3}_{\mathrm{South}}$ of the toy example 122
6.2 Flow graph $\mathcal{G}(f)$ of the toy example from Chapter 4 124
6.3 Organization of the three computation pipelines of a vertex 127
6.4 Computational blocks in the ACP of vertex $\mathrm{S3}_{\mathrm{South}}$ 127
6.5 Content of the ACP data structure at several observations points of Figure 6.1. 129

7.1 Principle of the event scheduler in a DES 139
7.2 Illustration of the solutions embraced by ns-2 and ns-3 to specialize the Nodes 140
7.3 Illustration of the solutions embraced by FP-TFA and xTFA for distinguishing between vertex models 141
7.4 Principles and issues of [Zhu, Ma, Ryu 2013] 143
7.5 UML class diagram of the design of local clocks in ns-3 145
7.6 Sequence diagram of the proposed design for the use-case “Scheduling a new Event” 147
7.7 Sequence diagram of the proposed design for the use-case “Updating the Clock Model” 149
7.8 Scenario showing the instability of the non-adapted IR in synchronized networks 150
7.9 Shape of the function $h_i(t)$ for the adversarial clock $\mathcal{H}_i$. 150
7.10 elay spent in the IR as a function of the packet number 151

8.1 Simplified physical topology of the Volvo core TSN Network 154
8.2 End-to-end latency bounds on the Volvo use-case. 155
8.3 End-to-end latency bounds on the Volvo use-case (continued) 157

2 Illustration de nos contributions théoriques 172

A.1 Model for any device in the time-sensitive network. 187
A.2 Relation between the bridges, the vertices of the underlying graph and the observation points 187

B.1 Illustration of the proof principle 189
B.2 Notations for Appendix B.2.1 191
B.3 Notations for Appendix B.2.2 193
B.4 Source output in the trajectory that achieves the tightness of Corollary 4.1 . 195
B.5 Example of the source output in the trajectory that achieves the tightness, focusing on categories I and B. 196
B.6 Source output, with the three categories I, B and S of data-units 197
B.7 Source output, with the four categories of data units: I, B, S and X of data-units, in the trajectory that achieves the tightness of Corollary 4.1. 199
B.8 Cumulative function R^* at the output of the PEF 203
B.9 Example of the source output in the trajectory that achieves the tightness, focusing on categories I and B. 208
B.10 Unfolding $\mathcal{N}$ into a feed-forward causal system. 211
B.11 Notations for the Proof of Theorem 4.4. 215
B.12 Generation of data units for flows f_i and f_{i+1} 218
B.13 Traffic profile for the flows f_i and f_{i+1} at the output of the two adversarial paths 218

B.14 Traffic profile for the flows f_i and f_{i+1} at the input of the IR and at its output 219
B.15 Traffic profile for the flows f_i and f_{i+1} at the output of the two adversarial paths 223
B.16 Case where the initial offset of the local clock $\mathcal{H}_i$ is positive compared to the reference clock $\mathcal{H}_g$. 226
B.17 Network-calculus model of the system S_k followed by the PFR REG$_k$ 231
B.18 TAI delay bound computation for the flow through the system S' of Figure B.17 231
B.19 Shape of the relative time function $d_{\text{PFR}\leftarrow\text{TAI}}$ 233
B.20 Delay-bound computation for the flow through the concatenation of S_k with the fluid greedy shaper 236
B.21 Shape of the time function $d_{j\leftarrow\text{IR}}$ 237
B.22 Generation of packets as observed with $\mathcal{H}_j$ 238
B.23 Generation of packets as observed with $\mathcal{H}_{\text{IR}}$ 238
B.24 Traffic arrival from two successive upstream sources, as observed with $\mathcal{H}_{\text{IR}}$ . . 239

List of Tables

2.1 Overview of the Literature On the Computation of Residual Service-Curves . 25

3.1 Summary of the Regulator Deployment Approaches 44

4.1 Main Acronyms Used in the Chapter and Comparison with the Terms of the Working Groups. 65
4.2 Overview of the Configuration Options supported by TSN FRER and DetNet PREOFs and their Analysis in the Literature. 67
4.3 Benefits and Drawbacks of Several Configurations, Compared to the Situation with the PEF(s) only. 86

5.1 Relations Between an Arrival Curve Observed with $\mathcal{H}_i$ and an Arrival Curve Observed with $\mathcal{H}_g$. 102
5.2 Relations Between a Service Curve of a System Observed with $\mathcal{H}_i$ and a Service Curve of the same System Observed with $\mathcal{H}_g$. 103

7.1 Comparison of the terms used in the thesis with the ns-3 terminology. 140

8.1 Description of the Flows in the Use-Case 154

9.1 Summary of The Contributions in This Thesis and Classification per Level . 164
1 Résumé des Contributions de la Thèse et Classification par Niveau. 176

A.1 List of Notations 186
A.2 List of Notations (continued) 188

B.1 Generation of the Data-Units of Category I in the Trajectory that Achieves the Tightness of Corollary 4.1. 195
B.2 Generation of the Data-Units of Category B in the Trajectory that Achieves the Tightness of Corollary 4.1. 196
B.3 Generation of the Data-Units of Category S in the Trajectory that Achieves the Tightness of Corollary 4.1. 197
B.4 Generation of the Data-units of Category X in the Trajectory that Achieves the Tightness of Corollary 4.1. 198
B.5 Absolute Release Time for Each Packet at the Output of Each System S_1, S_2 202
B.6 Generation of the Data-Unit of Category I in the Trajectory that Achieves the Tightness of Corollary 4.1, when $d_2 - D_1 \leq b/r$. 208
B.7 Generation of the Data-Units of Category B in the Trajectory that Achieves the Tightness of Corollary 4.1, when $d_2 - D_1 \leq b/r$. 208
B.8 Absolute Release Time for Each Packet at the Output of Each System S_1, S_2, in the Case $d_2 - D_1 \leq b/r$. 209
B.9 Notations for the Proof of Theorem 4.3 212

List of Publications

List of Publications

- Ludovic Thomas, Jean-Yves Le Boudec, and Ahlem Mifdaoui [Dec. 2019]. "On Cyclic Dependencies and Regulators in Time-Sensitive Networks." In: *2019 IEEE Real-Time Systems Symposium (RTSS)*. 2019 IEEE Real-Time Systems Symposium (RTSS), pp. 299–311. DOI: 10.1109/RTSS46320.2019.00035
- Ludovic Thomas and Jean-Yves Le Boudec [June 9, 2020]. "On Time Synchronization Issues in Time-Sensitive Networks with Regulators and Nonideal Clocks." In: *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 4.2, 27:1–27:41. DOI: 10.1145/3392145
- Ludovic Thomas, Ahlem Mifdaoui, and Jean-Yves Le Boudec [2022]. "Worst-Case Delay Bounds in Time-Sensitive Networks With Packet Replication and Elimination." In: *IEEE/ACM Transactions on Networking*, pp. 1–15. ISSN: 1558-2566. DOI: 10.1109/TNET.2022.3180763

List of Software Creations

- Ludovic Thomas [2022]. *THOMAS Ludovic / Xtfa*. ISAE-SUPAERO. URL: https://gitlab.isae-supaero.fr/l.thomas/xtfa [visited on 06/22/2022]
- Guillermo Aguirre Rodrigo and Ludovic Thomas [June 2020]. *Clock: Module for Local Clock Implementation (!332) · Merge Request · Nsnam / Ns-3-Dev*. GitLab. URL: https://gitlab.com/nsnam/ns-3-dev/-/merge_requests/332 [visited on 04/10/2022]

Introduction

Time-sensitive networks have been used since the 1990s in *cyber-physical* systems (e.g., cars, planes, factories) for interconnecting sensors, processing units, and actuators, and for enabling *safety-critical* applications such as control loops. Before 2010, time-sensitive networks were present only in a few industrial sectors (most notably the aerospace and car industries), and various time-sensitive networking technologies were used for the different sectors.

As opposed to the *best-effort* quality of service offered by non-safety-critical communication networks (such as the Internet), time-sensitive networks serve the data flows with a *deterministic* service that includes a guarantee of bounded latencies with no congestion losses. This “core” service is our main focus in this thesis. Several mechanisms, such as *schedulers*, *shapers* and *frame preemption* mechanisms, were developed for providing such service.

Proven latency and backlog bounds must be computed to validate the timing requirements and absence of congestion losses with these mechanisms. Several deterministic frameworks for computing such bounds were developed and are widely used to compute the effects of the above mechanisms on latency bounds.

Two main ongoing developments constitute the context of this thesis:

• First, there exists an emerging need for the standardization of time-sensitive networks. Many industrials have joined the time-sensitive networking (TSN) task group of the Institute of Electrical and Electronics Engineers (IEEE) or the deterministic networking (DetNet) working group of the Internet Engineering Task Force (IETF). Each group specifies a set of technologies that are independent from the industry sector. The industry sectors can then cherry-pick the technologies according to their specific needs.

• Second, time-sensitive networks are required to provide wider services. For example, IEEE TSN networks provide not only *bounded latency* but also a *high reliability*, a service of *time synchronization* and an easy reconfiguration with the use of multi-path topologies that provide alternative routes. New topologies (e.g., multi-path topologies) and new mechanisms (e.g., redundancy and synchronization), were developed to enable these additional services, and their ability to provide the aforementioned new services is studied in the literature.

However, the side effects of these new mechanisms on the *deterministic* service and their combinations with scheduling mechanisms are hardly studied in the literature. Furthermore, the existing deterministic frameworks do not consider these new topologies and mechanisms. As we demonstrate in this thesis, these frameworks must be adapted to the presence of the new mechanisms otherwise they can compute invalid performance bounds.

In this thesis, we analyze the side effects on latency bounds of the combinations of the new mechanisms (redundancy, time synchronization) and topologies (multi-path) by relying on network calculus. We provide the theoretical grounds for modeling the effects of the new mechanisms in the network-calculus deterministic framework. We also study the interactions with the traffic regulators and in particular with the interleaved regulator (IR) (implemented by IEEE TSN *asynchronous traffic shaping*) We show that the interactions between redun-

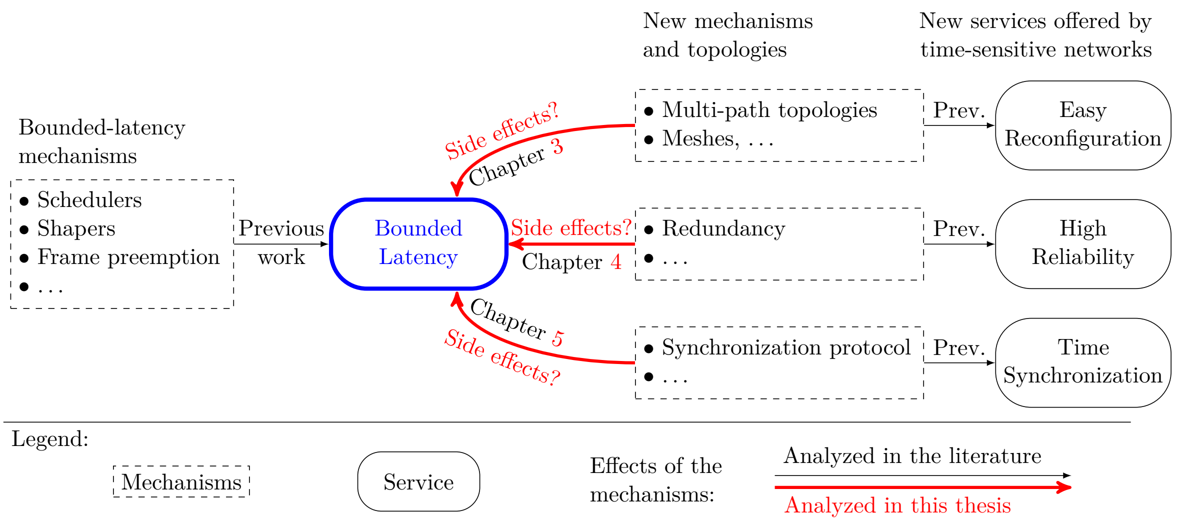


Figure 1: Illustration of the context of the thesis and of our theoretical contributions. The *bounded-latency* service in the thick-blue-lined oval is the focus of this thesis. Several *bounded-latency mechanisms* (dashed box on the left) were developed to achieve this service and are studied in numerous occasions in the literature. A new set of services was introduced in time-sensitive networks (ovals on the right), for which new mechanisms were developed. Their performance for their respective service is also studied in the literature. In this thesis, we study whether these new mechanisms and new topologies could cause side effects on the *bounded-latency* service. Figure based on [Farkas 2018, p. 5].

dancy, time-synchronization mechanisms and the interleaved regulator can yield unbounded latencies.

The thesis is organized as follows. In the first part, we introduce the context and provides the related work: In Chapter 1, we provide the technological context on time-sensitive networks and discuss their performance evaluation. In Chapter 2, we focus on the network-calculus framework. Our theoretical contributions are then presented in the second part: In Chapter 3, we analyze the effects of cyclic dependencies on latency bounds and their interactions with traffic regulators. In Chapter 4, we analyze the effects of redundancy mechanisms on latency bounds and their interactions with traffic regulators. In Chapter 5, we analyze the effects of an imperfect synchronization (or of its absence) on the latency bounds and its consequences for traffic regulators. Our practical contributions are presented in the last part: In Chapter 6, we describe experimental modular TFA (xTFA), a tool for computing latency bounds in time-sensitive networks that implement the theoretical results of this thesis. In Chapter 7, we describe an addition to the ns-3 network simulator; it enables us to simulate timing inaccuracies and the issues that they can cause with IRs. In Chapter 8, we present an application of our results to a representative industrial use-case. We provide our conclusive remarks and future research perspectives in Chapter 9.

Appendix A is intended to be a *vade mecum* for the manuscript and can be printed separately: it provides the list of acronyms, the table of notations and a glossary. Throughout the thesis, the underlined terms are defined in the glossary. Appendix B contains the proofs of the theoretical results.

Part I

Context

Chapter 1

Technological Context: Time-Sensitive Networks

Curiosity. Insight. Spirit. Opportunity. [...] If rovers are to be the qualities of us as a race, we missed the most important thing: Perseverance. [...] We, [...] as humans, will not give up.

Alex Mather, Essay to name the NASA Mars 2020 rover.

Contents

1.1 Service Performance of Communication Networks 6
1.1.1 Performance Metrics of Communication Networks 6
1.1.2 Quality of Service 6
1.2 Technological Trends in Time-Sensitive Networks 7
1.2.1 Taxonomy of Time-Sensitive Networks 7
1.2.2 Trends in the Use of Time-Sensitive Networks 8
1.3 The IEEE TSN Task Group 10
1.3.1 Content of the TSN Activities 10
1.3.2 Organization of the TSN Documents 11
1.3.3 Difference between TSN and other Ethernet-based Technologies 12
1.4 The IETF DetNet Working Group 12
1.5 Performance Analysis of Time-Sensitive Networks 12

The functionalities of most human-made complex systems (e.g., cars, airplanes, web-services) are achieved by exchanging pieces of information (the data units) among the system elements through one or several communication networks. For example, an autonomous car has several control loops in which data is gathered from the sensors and is distributed to one or several controllers; the controllers then interact to make a control decision that is then sent to the actuators. The communication network provides the service of transporting the data units from a sending application in an end system to one or several receiving applications in one or several end systems. Understanding the performance of the service provided by the communication network is fundamental for obtaining system-wide performance metrics and for validating the system requirements.

In particular, if a communication network supports a safety-critical application (*i.e.*, an application whose failure can result in death or serious injuries to people, significant economical loss, or harm to the environment [INCOSE SE Handbook, §10.10]), then the safety requirements at the system level derive requirements on the guarantees offered by the network's service. Time-sensitive networks are a subset of communication networks specifically designed for providing a service with deterministic guarantees to safety-critical applications. They are at the core of several multi-stakeholder working groups, such as IEEE TSN and IETF DetNet, and they constitute the technological context of this thesis.

In this chapter, we first present the performance metrics of interest for a time-sensitive network, and we define the deterministic service in Section 1.1. We then give a brief overview of the industrial trends and provide some elements of taxonomy in Section 1.2. Afterwards, we introduce more specifically the IEEE TSN task group (Section 1.3) and provide some remarks on IETF DetNet (Section 1.4). Last, we give an overview in Section 1.5 of the related work on the performance analysis of time-sensitive networks.

1.1 Service Performance of Communication Networks

1.1.1 Performance Metrics of Communication Networks

In a communication network, the data units are organized into flows: a flow is a coherent sequence of data units that originate at a source and traverse the network to reach one or several destination(s). The service offered by a given communication network can be evaluated by a variety of functional performance metrics. In the thesis, we use

- the latency of a data unit is the duration needed by the data unit to travel the network from source to destination(s). The latency of a flow is the maximum latency of any of its data units.
- the jitter of a flow represents the difference between the maximum and the minimum latency of any of its data units.
- the packet loss ratio of a flow quantifies the fraction of its sent data units that are lost or altered in the network.
- the packet-reordering metrics for a flow [Mohammadpour, Le Boudec 2021] quantify the amount of out-of-order data units at one of its destinations.

The network is also characterized by a set of non-functional properties such as its cost, its weight, its flexibility (ability to add/remove/change flows or network elements), its maintainability, its scalability, etc.

1.1.2 Quality of Service

The Quality of Service (QoS) characterizes the guarantees that a network provides on its functional performance metrics. For example, when the network does not provide any guarantees for the latency, jitter, loss-ratio or the mis-ordering of a flow, then we say that the network provides a best-effort service. When the network provides guarantees on the worst-case performance metrics, we say that the network provides a deterministic service.

Time-sensitive networks provide a deterministic service to the applications. The focus on the worst-case performance metrics is explicit in the working groups, including IEEE TSN [Farkas 2018] and IETF DetNet [RFC 8655],[48].

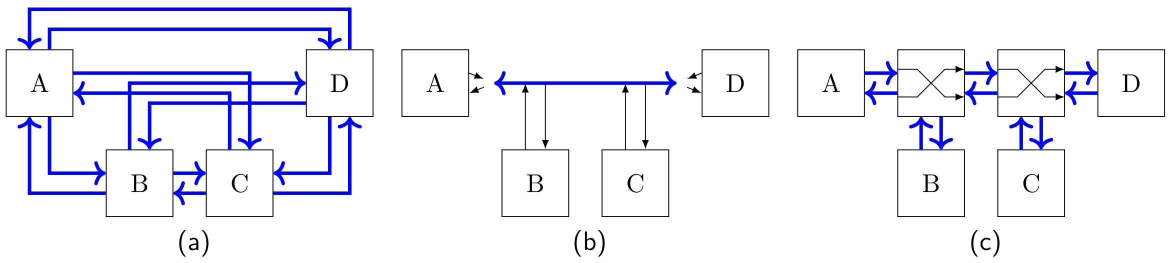


Figure 1.1: Evolution of the paradigms used in time-sensitive networks for sharing the network resources between the end systems (boxes in the figure). (a) With the point-to-point paradigm, one transmission link for each pair of end systems and for each direction is used (a total of 12 links in this figure). (b) With the bus paradigm, only one transmission link is used and all the end systems are connected to this link. (c) In the switched paradigm, transmission links are used for connecting the end systems to switches and between the switches. This example requires two switches and 10 transmission links. Here, the main shared resource are the switches, which is the focus of this thesis.

The notion of deterministic service should not be confused with the notion of predictable behavior: A network has a predictable behavior if its state at any time instant can be predicted from the properties of the network. A network with a predictable behavior provides a deterministic service: the worst-case performance metrics can be predicted. However, there exist non-predictable networks that provide also a deterministic service[1].

1.2 Technological Trends in Time-Sensitive Networks

1.2.1 Taxonomy of Time-Sensitive Networks

Time-sensitive networks interconnect the end systems and the applications that they contain by relying on the transmission links (*i.e.*, wires) provided by the physical layer of the layered model. Each transmission link represents an associated cost (e.g., hardware cost, weight). The most common paradigms for sharing the network resources are presented in Figure 1.1.

Accessing the Shared Resources: Synchronous, Asynchronous

In the *bus* and in the *switched* paradigms (Figure 1.1b and 1.1c), some resources of the network are shared between the end systems: the transmission link itself for the bus paradigm and the switches for the switched paradigm. Access to these shared resources must be distributed among the applications. There exist two main modes for distributing the access:

Time-Triggered Sources in Synchronous Networks: A synchronous network uses the Time-Division Multiple Access (TDMA) mode: the sources have access to the resources at

[1]To follow up on a question that we received during our defense and on subsequent discussions that arose, we would like to emphasis here that we do not use the term *determistic network*, *i.e.*, we do not use the adjective *determistic* to describe the *network*. The adjective *deterministic* is here only used for the *service*. This avoids to take part in the debate around the correct definition for *determistic network*. The only terms that are used in this manuscript are *deterministic service* (as defined by IEEE TSN and IETF DetNet) and *predictable behavior*, for which we believe that we provide an appropriate definition (the behavior can be predicted). The definition of *deterministic network* is left open for the reader.

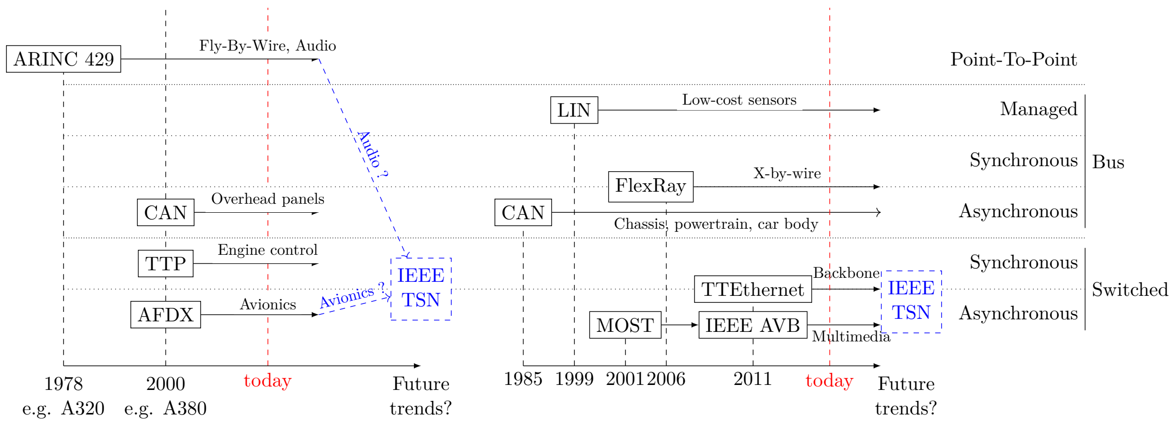


Figure 1.2: Classification of the past and future trends of time-sensitive networks in two industrial sectors: the aerospace industry (on the left) and the automotive industry (on the right). In today's systems, we note the coexistence of low-capacity buses and high-capacity Ethernet-based networks. The use of the former is expected to continue, whereas the latter are expected to merge towards IEEE TSN.

predetermined time instants, based on a global schedule that is shared across the network. The emission of the data units is *time triggered*, *i.e.*, triggered when the node's internal clock reaches the allowed transmission slot in the global schedule.

Synchronous networks exhibit a predictable behavior hence provide a deterministic service by nature. The complexity of this approach resides in the computation of a global schedule that meets all the applications' constraints. This problem is NP-complete [Raagaard, Pop 2017]. Synchronous networks are also required to be time synchronized: a common notion of time must be shared across all the nodes in the network.

Event-Triggered Sources in Asynchronous Networks: In an asynchronous network, the applications do not need to wait for predetermined time instants to send their data units. The emission of the data units is *event triggered*, *i.e.*, triggered by an external event, such as an action of the environment on one of the end-system's sensors. Asynchronous networks are unpredictable by nature, hence proving their deterministic service is challenging.

Asynchronous networks are typically not required to be time synchronized, but time synchronization can be present for other purposes (network management, time stamping).

Managed Buses: Some types of buses use a third mode in which a unique bus manager initiates the conversations on the bus. This is also called the master/slave mode.

1.2.2 Trends in the Use of Time-Sensitive Networks

Figure 1.2 compiles some overall trends in the use of time-sensitive networks in two different industrial sectors: the aerospace industry (commercial airplanes) and the automotive industry (commercial cars). For the aerospace industry (on the left), the past trends are compiled from the lecture of Prof. Mifdaoui at *Institut Supérieur de l'Aéronautique et de l'Espace* (ISAE), whereas the future trends constitute our own understanding as intuited from the IEEE TSN aerospace profile [IEEE P802.1DP] and from the presentation of the EDEN project at *Ecole* *d'été Temps-Réel 2021* (ETR21) [Cuenot 2021]. EDEN is a multi-stakeholder project for enabling the deployment of IEEE TSN as an embedded network in multi-domain architectures. For the automotive industry (on the right), the past trends are compiled from [Navet, *et al.* 2005; Navet, Simonot-Lion 2013; Wilwert, *et al.* 2005] and the future trends constitute our own understanding as intuited from the IEEE TSN automotive profile [IEEE P802.1DG] and again from the presentation of the EDEN project [Cuenot 2021]. Due to the complexity of the systems in both industrial sectors, many more technologies were developed than those reported in Figure 1.2, but not all technologies have been effectively used in production.

Among those reported in the figure, we note a few synchronous/asynchronous hybrid networks that merge the benefits of both modes (for example running control loops with time-triggered sources while transporting also alert messages from event-triggered sources).

Despite the diversity of the technologies, we also note a general trend that every system uses two different types of networks: low-performance networks based on point-to-point or bus architectures and emerging high-performance networks based on switched Ethernet.

The bus-based and point-to-point-based technologies developed before 1999 are used mostly at the edges of the systems, *i.e.*, for connecting large quantities of sensors and actuators that do not require large volumes of data exchange. The use of these technologies is expected to continue in the future. Indeed, the non-functional requirements (cost, scalability to large numbers of sensors, compatibility with legacy devices, familiarity of the industrials, etc.) outweigh the performance requirements (sensor data volume), hence the use of high-performance Ethernet-based protocols is not justified.

High-performance Ethernet-based protocols are primarily developed to support new complex functionalities that require large volumes of data. For example, the self-driving function of an autonomous car requires the processing of pictures, telemetry maps, etc. Hence, these new networks are focused at the core of the system, in connection with a few high-volume sensors (such as cameras). Since the development of FlexRay, these new technologies often rely on an hybrid access mode.

Many of the new high-speed time-sensitive networking technologies rely on the IEEE Ethernet specifications ([IEEE 802.1Q] and [IEEE 802.3]). This choice is based on the predominance of Ethernet in the world of computer networking. For example, using the Ethernet-based AFDX protocol in the aerospace industry (a) saves formation costs, because most engineers know about Ethernet, (b) saves costs for the test infrastructures, because many tools for debugging, monitoring and simulating Ethernet networks already exist, and (c) prevents the suppliers of AFDX devices (switches and end systems) from enjoying from a monopolistic position, because less effort is required for a concurrent to adapt. These arguments and many more can be described as "Everyone knows about Ethernet".

In this thesis, we focus on the second type of time-sensitive networks, *i.e.*, those based on the switched Ethernet. As we discuss in the next section, these technologies are expected to converge towards the mechanisms standardized by IEEE TSN that constitute, therefore, the primary context of this thesis.

1.3 The IEEE TSN Task Group

Time-sensitive networking (TSN) is a task group of the Institute of Electrical and Electronics Engineers (IEEE); it provides a set of mechanisms and specifications for designing time-sensitive networks based on IEEE local area networks (LANs). The task group is chartered to provide deterministic service with guaranteed bounded latency, low jitter, zero congestion-loss, and a low packet-loss ratio [21]. The group was formed in 2012 and relies on the former IEEE Audio Video Bridging (AVB) task group, which focused on Ethernet networks for audio and video equipment.

1.3.1 Content of the TSN Activities

The TSN task group produce two main types of outputs.

A Toolbox of Features

The most important production of the task group is a set of technological features that define a switched network that can support both synchronous and asynchronous access modes. The service offered by TSN is described by four key components [Farkas 2018]:

- A deterministic service in the form of **bounded low latency** and zero loss by congestion: TSN provides a set of features that augment the traditional forwarding process of an Ethernet bridge in order to allow for bounded latency. The toolbox contains synchronous schedulers for time-triggered sources (*Enhancements for Scheduled Traffic* [IEEE 802.1Qbv], *Cyclic Queuing and Forwarding* [IEEE 802.1Qch]), and asynchronous class-based schedulers for event-triggered sources (*Credit Based Shaper* [IEEE 802.1Qav]), as well as interleaved regulators (*Asynchronous Traffic Shaping* [IEEE 802.1Qcr]).
- A service of **time synchronization**: TSN specifies a time-synchronization protocol [IEEE 802.1AS]. Time synchronization is provided to the applications (e.g., for time-stamping sensor data) and is required when using the synchronous features of TSN.
- A high level of **reliability**: TSN provides a set of features that improve the reliability of the network. This includes protection against babbling idiots through traffic policing (*Per-Stream Filtering and Policing* [IEEE 802.1Qci]), as well as redundancy (*Frame Replication and Elimination for Reliability* [IEEE 802.1CB]).
- A simplified network management with an **easy reconfiguration** and the support of multi-path topologies that reduce the reconfiguration effort.

Depending on the system's requirements, the network engineer then selects the most suitable features among the TSN toolbox. We note that some features (e.g., the schedulers) are selected on a per-class, per-output-port, per-bridge basis, whereas others (e.g., the redundancy) are selected on a per-flows basis[2]. As a consequence, finding the best set of features

[2]At the time of this writing, the TSN documents also prevent several combinations of features, even if there exist no fundamental reason for these limitations. For example the current version of [IEEE 802.1Qcr] and [IEEE 802.1Q] prevent the combination of asynchronous traffic shaping (ATS) and credit-based shaper (CBS)

and their parameters for a given set of requirements remains an open issue [NAVET, MAI, MIGGE 2019].

For each of the four above objectives, we typically expect that a theory can be selected to validate that the mechanisms labeled with this objective indeed provide the required objective level. In Section 1.5, we discuss some of the available frameworks for validating the effect of the *bounded-latency* mechanisms on the bounded latency. The validation of the time-synchronization objective based on the performance of the time-synchronization protocol will typically use the tools from the time-metrology domain whereas the validation of the reliability objective will typically use the tools from the reliability-analysis domain.

However, it appears that the cross-interactions (effect of the mechanisms developed for an objective A on an objective B) are hardly discussed in the TSN task group.

A Set of Application Profiles

To ease the network design, the TSN task group also provides a set of TSN *profiles*. A profile is a document that lists the common assumptions and requirements for a given industrial application field. It also lists recommended architectural choices, recommended TSN features for the given application field, and recommended parameters for these features.

At time of this writing, five different TSN profiles for various industrial applications were published or are ongoing projects: aerospace [IEEE P802.1DP], automotive [IEEE P802.1DG], professional audio and video [IEEE 802.1BA], industrial automation [IEC/IEEE 60802] and cellular front-haul networks [IEEE 802.1CM].

1.3.2 Organization of the TSN Documents

The task group works on two different types of documents. All documents start with 802.1, and the ongoing projects start with a leading "P".

– The base standards are stand-alone documents. They end with capital letters[3]. In TSN, the main base standard is [IEEE 802.1Q], that specifies the bridges. The TSN synchronization feature is described in the independent [IEEE 802.1AS] document, and the redundancy feature is described in [IEEE 802.1CB]. Base standards are periodically updated with accepted amendments (see below).

– The amendments are proposed modifications to a base standard. For example, most TSN features for bounded latency propose improvements of the forwarding process within bridges, they thus amend the [IEEE 802.1Q] specifications. They end with lowercase letters[4]. For example, the ATS feature is specified in the [IEEE 802.1Qcr] amendment. Once accepted, the amendments are periodically merged into the base standards.

for the same class, because only one transmission selection algorithm can be selected per class. Yet combining ATS and CBS presents an interest for the performance bounds, as outlined by the different studies that analyze their combination [Mohammadpour, *et al.* 2018; Zhao, Pop, Steinhorst 2021]. As this technical limitation might be corrected in the future, we consider in this thesis that the interleaved regulator (the model behind TSN ATS) can be combined with class-based schedulers with no limitations.

[3]Letters are assigned in chronological order: 802.1AA comes just after 802.1Z, and so on.

[4]Letters are assigned in chronological order: 802.1Qaa comes just after 802.1Qz, and so on.

1.3.3 Difference between TSN and other Ethernet-based Technologies

In Section 1.2.2, we mention that the "Everyone knows about Ethernet" principle reduces the costs associated with the development of Ethernet-based time-sensitive networks in the aerospace and the automotive industries. However, the different Ethernet-based time-sensitive networking technologies have, so far, been limited to one or a few industrial sectors. For example, the Avionics Full-dupleX switched Ethernet (AFDX) is not used in any cars. Hence, there exist remaining cost and monopoly issues in the gap between Ethernet and the industry-specific network such as AFDX.

By segregating the context-agnostic technologies in the toolbox (Section 1.3.1) from the industry-specific profiles (Section 1.3.1), the idea of the IEEE TSN task group is to go even further with an "Everyone knows about TSN" principle. Hence, the product manufacturers, the engineers and the debugging tools can serve any industry sector. And the specificity of a sector lies solely on the corresponding IEEE TSN *profile* document. Therefore, it appears that all Ethernet-based technologies will be replaced in the future by IEEE TSN. The task group already gathers the main actors of the different sectors.

However, the technologies developed by the task group are much more complex and diverse than those developed, for example, in AFDX or TTEthernet. This complexity represents a potential risk that can hinder the adoption of TSN across the industry sectors: It might motivates the manufacturers to focus on a few TSN profiles, implement only the technologies listed in these profiles and neglect the other features.

1.4 The IETF DetNet Working Group

Deterministic networking (DetNet) is a working group of the Internet Engineering Task Force (IETF); it focuses on the same aspects as IEEE TSN, *i.e.*, providing a deterministic service, but for applications that span over the third layer of the Internet's layered model. In fact, DetNet collaborates with IEEE TSN to define a common architecture of time-sensitive networks for both Layer 2 and Layer 3. Note, however, that DetNet focuses only on layer-2 and layer-3 networks that are under a single administrative control; its does not consider the Internet at large.

The activities of DetNet focus on: defining the overall architecture and functions; defining the identification and processing of DetNet flows over the MPLS and IP protocols; defining the relevant data, models and control-plane solutions for deploying DetNet; and providing recommendations on the computation of latency bounds. Note that the actual allocation of the underlying resource to the flows is not standardized by DetNet, but delegated to the lower-layer (*i.e.*, IEEE TSN).

1.5 Performance Analysis of Time-Sensitive Networks

Figure 1.3 illustrates the distribution of the latencies in a network. When the network is a public network that does not support safety-critical applications or when the network supports *soft* real-time applications (*i.e.*, non-safety-critical applications that can tolerate packets received after the deadline), then the performance of the network is often evaluated based

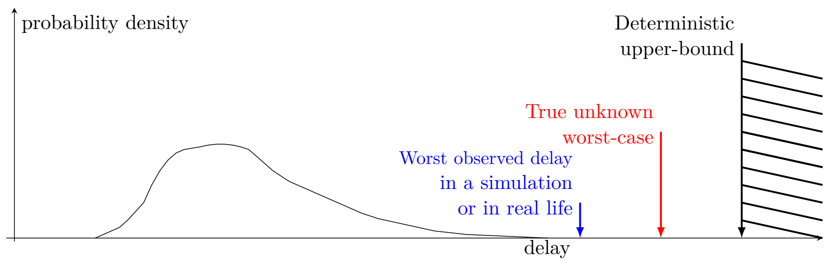


Figure 1.3: Illustration of the distribution of the latency in a network. The value of worst-case latency remains in general unknown. Simulations and measurements can only provide lower-bounds of the unknown worst-case. Deterministic approaches can provide an upper-bound. This upper-bound can the be used to validate the performance requirements of the network.

on its stochastic properties such as its mean, its distribution, and sometimes the shape of its distribution tail [Song, Koubaa, Simonot 2002].

However, when the quality of service is deterministic, the network must provide deterministic guarantees on the worst-case latency. Therefore, time-sensitive networks that support safety-critical applications and that provide this deterministic service cannot be analyzed using stochastic methods, as rare events are not captured by stochastic metrics.

Simulation or real-life measurements can report achievable latencies, *i.e.*, latencies that lower-bound the unknown worst-case; whereas, to validate the deterministic service, we require an upper-bound on the worst case. Obtaining such an upper bound requires the use of deterministic approaches. We provide an overview of some deterministic approaches used for time-sensitive network; note that the below list is not intended to be exhaustive.

Model Checking

The model-checking approach [Clarke, Emerson, Sistla 1986] describes a system as a set of states and transitions. Once the state-machine of the system has been obtained, model checking can be used for verifying the properties of a system. For example, we can verify that a system never enters an unsafe state or a blocking state. Several widely spread commercial tools for model-based system engineering rely on the model-checking approaches [2].

Model-checking was used to verify the timing properties of several time-sensitive network technologies [Krakora, *et al.* 2004], including Ethernet [Witsch, *et al.* 2006]. Model-checking approaches provide the exact worst-case latency, but they do not scale to complex architectures, because the number of possible states of the network increase significantly with its size.

Trajectory Approaches

The trajectory approach [Martin 2004] consists in modeling the worst interference that a flow can suffer from all the other interfering flows. The complexity of the approach resides in the complexity of modeling large numbers of interference patterns and various network systems. In [Li, Cros, George 2014], an error on the modeling of the serialization effect in the trajectory approach is described and a subsequent correction is proposed.

The approach was used for computing the worst-case bounds for the AFDX network in [Bauer, Scharbarg, Fraboul 2010; Li, Cros, George 2014]. An approach similar to the trajectory approach is proposed for TSN in [Docquier, *et al.* 2020]. Last, we note that the trajectory approach was also used to prove the *shaping-for-free* property of the Urgency Based Scheduler (the former name of TSN ATS) in [Specht, Samii 2016].

Holistic Schedulability Analysis

In the field of real-time scheduling on single-core and multi-core systems, the schedulability analysis discusses whether a set of periodic or aperiodic *tasks* can share one or several common resources (the *processors*) while meeting their deadlines [Liu, Layland 1973]. The holistic schedulability approach [Tindell, Clark 1994; Pop, Eles, Peng 1999] relies on these ideas for a set of end systems interconnected by shared buses: "*communication tasks are scheduled on buses similar to the way processes are scheduled on programmable processors*" [Eles, *et al.* 2000].

The theory has been applied for asynchronous buses, such as Controller Area Network (CAN) in [Tindell, Hansson, Wellings 1994], and synchronous buses, such as the Time Triggered Protocol, in [Eles, *et al.* 2000] for non-preemptive tasks and in [Pop, Eles, Peng 2004] for preemptive tasks.

The holistic schedulability analysis requires modeling the worst-case traversal time of a packet from the sending application to the receiving application, including any waiting time for accessing the bus. The worst-case traversal time of simple access policies, such as the non-preemptive static priority access of the CAN bus, can be modeled easily.

In [Pop, Eles, Peng 2003; Pop, *et al.* 2005], the theory is extended to buses connected via gateways. The theory was latter applied to compute the global schedule of TTEthernet, an hybrid synchronous/asynchronous switched network in [Tamas-Selicean, Pop, Steiner 2012; Tămaş–Selicean, Pop, Steiner 2015] and for the global schedule of IEEE TSN in [Pop, *et al.* 2016]. A major challenge that faces the holistic schedulability analysis for switched networks is that the worst-case delay of event-triggered messages is hard to determine across multiple hops and under various asynchronous scheduling policies (deficit round-robin, credit-based shaper, etc).

Network Calculus

Network calculus is a theory proposed by Le Boudec [Le Boudec 1996] for modeling network elements as in traditional system theory, with an input, an output, and a transfer function. The input/output representation makes it particularly useful for modeling switched networks with complex topologies. As we discuss in Chapter 2, a wide range of algorithms rely on the theory, some of which scale linearly with the size of the network. Network calculus provides upper bounds on the latency, buffer occupation, and burstiness of the flows in a network. Due to these properties, we select network calculus as the deterministic framework used in this thesis. We detail the framework in the next chapter.

Conclusion

Through the analysis of the industrial trends of time-sensitive networks, we have identified that the future safety-critical systems are expected to rely on two different types of networks: (1) buses that support applications with low data-volume requirements but stringent non-functional requirements and (2) switched-Ethernet networks that support new safety-critical applications with important data-volume requirements.

The latter technologies are expected to converge towards the unified sector-independant technologies developed by the IEEE TSN task group and by the IETF DetNet working group. Both aim at providing a deterministic service defined by the guarantees on the worst-case latencies offered to the flows and no losses by congestion.

To validate the deterministic service, the performance of such a network must be evaluated using deterministic analytical approaches. Among these approaches, the *model checking* approach, the *trajectory approach* and the *hollistic schedulability analysis* provide tight delay bounds for small networks (or buses). However, they do not scale well to complex switched networks.

Therefore, we chose to analyze IEEE TSN and IETF DetNet networks by using the network-calculus framework that we present in the following chapter.

Chapter 2

Theoretical Context: Network Calculus

"I have a file with 900 pages of analysis and contingency plans for war with Mars [...]. My file for what to do if an advanced alien species comes calling ? It's three pages long, and it begins with Step 1: Find God."

Chrisjen Avasarala

Mark Fergus and Hawk Ostby. *The Expanse* (TV series).
Based on the series of novels by James S.A. Corey.

Contents

2.1 The Underlying Formalism: Min-Plus Algebra 18
2.1.1 The Min-Plus Functions $\mathfrak{F}$ and the Curves $\mathfrak{F}_0$ 18
2.1.2 The Min-Plus Convolution $\otimes$ and the Min-Plus Deconvolution $\oslash$ 19
2.2 Main Concepts of Network Calculus 20
2.2.1 Cumulative Functions 20
2.2.2 Backlog and Virtual Delay 20
2.2.3 Arrival Curves 21
2.2.4 Service Curves 22
2.3 The Main Results of the Network-Calculus Theory 22
2.3.1 The Network-Calculus Three-Bound Theorem 22
2.3.2 Concatenation of Systems 23
2.3.3 Greedy Shapers, Packetizers 23
2.4 FIFO-per-Class Systems 24
2.4.1 Residual Service-Curves for Scheduling Policies Other than FIFO 25
2.4.2 The FIFO Scheduling Policy 25
2.5 Compositional Approaches for the Analysis of Feed-Forward FIFO-per-class Networks 26
2.5.1 Total Flow Analysis 27
2.5.2 Separated Flow Analysis 27
2.5.3 Tandems and Pay Multiplexing Only Once 28
2.5.4 Linear Programming 29
2.6 Network-Calculus Model For Time-Sensitive Networks 30
2.6.1 The Sources 30

2.6.2 The Input Port 31
2.6.3 The Switching Fabric 31
2.6.4 The Output Port 31
2.6.5 The Transmission Links 33
2.6.6 The Graph Induced By Flows, Flow Graph 33
2.6.7 Observation Points 34

Network calculus is a theory for obtaining bounds on the worst-case performance metrics of communication networks. The framework was developed by Le Boudec in [Le Boudec 1996] (and concurrently by Chang in [Chang 1997]), based on the seminal work of Cruz [Cruz 1991a; Cruz 1991b]. The theory now relies on several books of reference [Chang 2000; Le Boudec, Thiran 2001; Bouillard, Boyer, Le Corronc 2018] and tutorials [12; 13].

In this framework, the network elements are represented as in traditional system theory, with an input (α, the input *arrival curve*), an output (α^*, the output *arrival curve*), and a transfer function (β, the *service curve* of the network element) and the notions organize themselves around an algebraic structure: the min-plus dioid.

In this chapter, we provide some background on *deterministic* network calculus (simply denoted *network calculus* in this thesis). In Section 2.1, we present the min-plus dioid and the min-plus functions. Then, we present the main network-calculus notions in Sections 2.2 and its main results in Section 2.3. We also present the related work on FIFO-per-class networks in Section 2.4 and the compositional approaches for such networks in Section 2.4. Last, we discuss the network-calculus modeling of time-sensitive networks (IEEE TSN and IETF DetNet) in Section 2.6.

2.1 The Underlying Formalism: Min-Plus Algebra

Network calculus relies on the commutative dioid $(\mathbb{R}^+ \cup \{+\infty\}, \wedge, +)$, where $\mathbb{R}^+$ denotes the set of positive real numbers, $\wedge$ denotes the binary minimum between two real numbers and $+$ is the sum. For an element c of the min-plus dioid, the notation $|c|^+$ describes[1] $|c|^+ \triangleq \max(0, c)$.

2.1.1 The Min-Plus Functions $\mathfrak{F}$ and the Curves $\mathfrak{F}_0$

Definition 2.1 (Min-Plus Functions $\mathfrak{F}$ and Curves $\mathfrak{F}_0$, Le Boudec, Thiran 2001, §3.1.3) *We define the set of min-plus functions $\mathfrak{F}$ as the set of wide-sense increasing and positive functions $\mathfrak{f} : \mathbb{R} \to \mathbb{R}^+ \cup \{+\infty\}$ such that $\forall t < 0, \mathfrak{f}(t) = 0$. We define the set of* <u>*curves*</u> *$\mathfrak{F}_0$ as the subset of $\mathfrak{F}$ such that $\forall \mathfrak{f} \in \mathfrak{F}_0, \forall t \leq 0, \mathfrak{f}(t) = 0$.*

The functions of $\mathfrak{F}$ are assumed to be left continuous. The effects of this assumption are discussed in [Le Boudec, Thiran 2001, §1.1], [Boyer, Dufour, Santinelli 2013]. Figure 2.1 gives some examples of curves (*i.e.*, functions belonging to $\mathfrak{F}_0$) that are often used in this thesis.

[1]Throughout this thesis, we use the conventions on the $+\infty$ object: $\forall a \in \mathbb{R}, a + (+\infty) = (+\infty)$; $(+\infty) + (+\infty) = (+\infty)$; $a \wedge (+\infty) = a$; $(+\infty) \wedge (+\infty) = (+\infty)$; $\max(a, +\infty) = +\infty$.

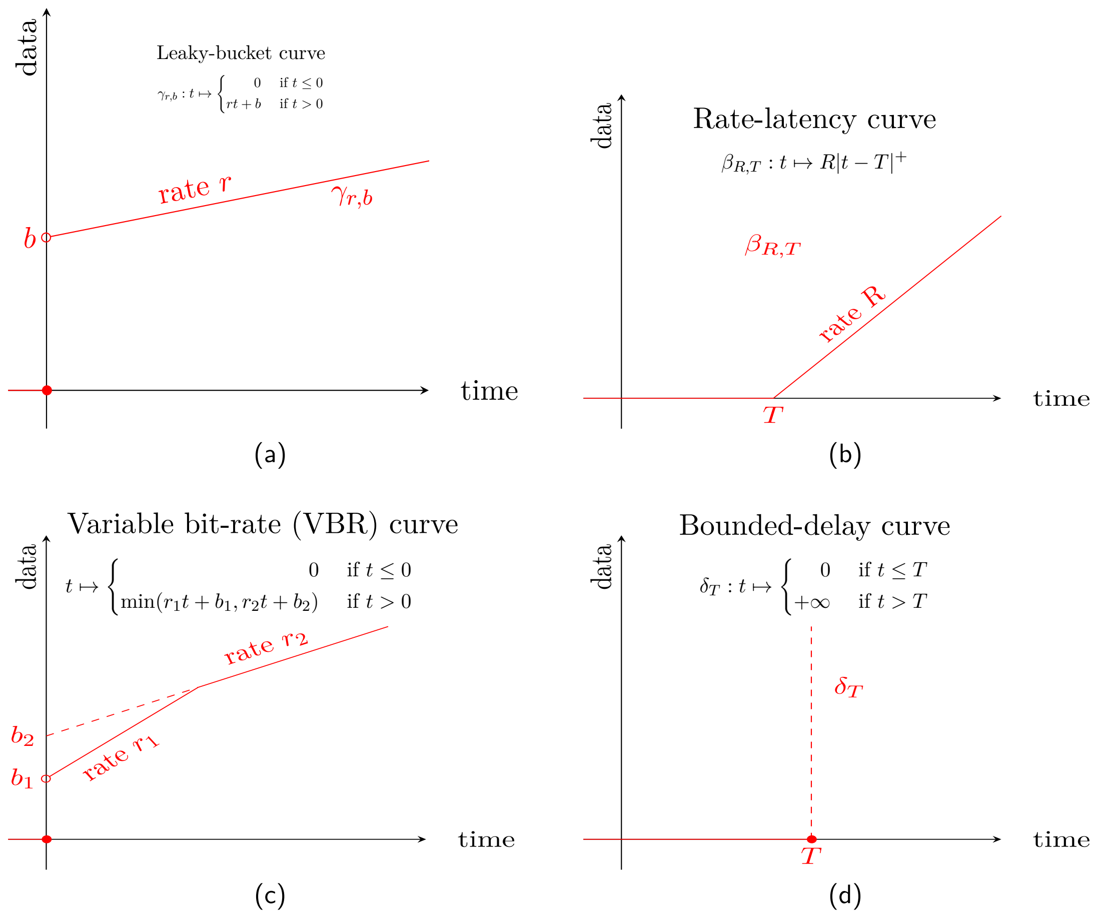


Figure 2.1: Example of curves used in this thesis. (a) Leaky-bucket curve with a rate r and a burst b. (b) Rate-latency curve with a rate R and a latency T. (c) Variable bit-rate (VBR) curve that can be written as the minimum of two leaky-bucket curves of rate r_1 [resp. r_2] and of burst b_1 [resp. b_2.] (d) Bounded-delay curve with a delay T.

2.1.2 The Min-Plus Convolution $\otimes$ and the Min-Plus Deconvolution $\oslash$

Definition 2.2 (Min-plus Convolution $\otimes$) *For $\mathfrak{f}, \mathfrak{g} \in \mathfrak{F}$, the min-plus convolution of $\mathfrak{f}$ and $\mathfrak{g}$ is $(\mathfrak{f} \otimes \mathfrak{g}) : t \mapsto \inf_{0 \leq s \leq t}\{\mathfrak{f}(t-s) + \mathfrak{g}(s)\}$.*

The min-plus convolution $\otimes$ is associative and commutative. Its neutral element is δ_0 (Figure 2.1d with $T = 0$). Its counterpart is the min-plus deconvolution.

Definition 2.3 (Min-plus deconvolution $\oslash$) *For $\mathfrak{f}, \mathfrak{g} \in \mathfrak{F}$, the min-plus deconvolution of $\mathfrak{f}$ by $\mathfrak{g}$ is $(\mathfrak{f} \oslash \mathfrak{g}) : t \mapsto \sup_{u \geq 0}\{\mathfrak{f}(t+u) - \mathfrak{g}(u)\}$.*

Proposition 2.1 (Classic convolutions. Le Boudec, Thiran 2001, Thms 3.1.4, 3.1.6)
- *If $\mathfrak{f}$ and $\mathfrak{g}$ are concave functions of $\mathfrak{F}_0$, then $(\mathfrak{f} \otimes \mathfrak{g}) = (\mathfrak{f} \wedge \mathfrak{g})$ with $\wedge$ the minimum.*
- *If $\mathfrak{f}$ and $\mathfrak{g}$ are piecewise-linear convex functions of $\mathfrak{F}_0$ then $\mathfrak{f} \otimes \mathfrak{g}$ is obtained by putting side-by-side the different linear pieces of $\mathfrak{f}$ and $\mathfrak{g}$, by order of increasing slope.*

Example:
- For any pair γ_{r_1,b_1}, γ_{r_2,b_2} of leaky-bucket curves, $\gamma_{r_1,b_1} \otimes \gamma_{r_2,b_2} = \gamma_{r_1,b_1} \wedge \gamma_{r_2,b_2}$ is a

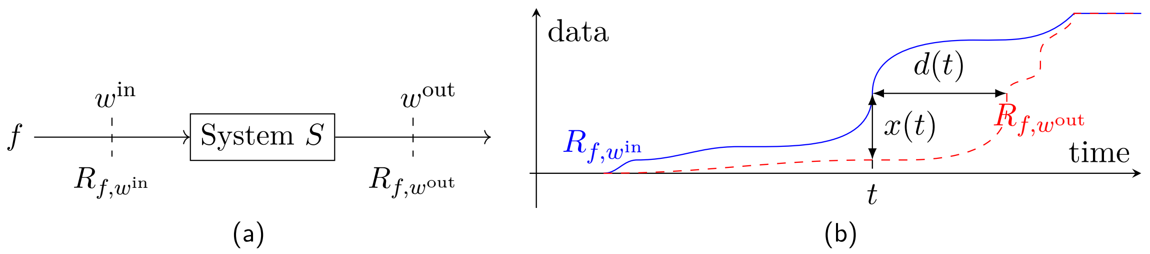


Figure 2.2: A flow f crosses a system S. (a) The observation point w^{in} [resp., w^{out}] is located at the input of the system [resp., at its output]. (b) Example of two cumulative functions for the flow f at the two observations points w^{in}, w^{out}.

variable-bit-rate (VBR) arrival curve (Figure 2.1c).
- For any pair β_{R_1,T_1}, β_{R_2,T_2} of rate-latency curves $\beta_{R_1,T_1} \otimes \beta_{R_2,T_2} = \beta_{(R_1 \wedge R_2),(T_1+T_2)}$.

Proposition 2.2 (Classic deconvolutions. Le Boudec, Thiran 2001, §3.1.9)
- *For any* $\gamma_{r,b}$,$\beta_{R,T}$, $\forall t > 0, (\gamma_{r,b} \oslash \beta_{R,T})(t) = \gamma_{r,b+rT}(t)$
- *For any* $\mathfrak{f} \in \mathfrak{F}_0$, *for any* $D > 0$, $(\mathfrak{f} \oslash \delta_D) : t \mapsto f(t + D)$.

In [Bouillard, Boyer, Le Corronc 2018, Chap. 4], the choice of a family of min-plus functions closed by the min-plus operations is discussed and efficient algorithms are provided. Similar discussions can be found in [Schmitt, Zdarsky 2006, §2]. An overview of the tools that provide a min-plus back-end is available in [Bouillard, Boyer, Le Corronc 2018, §4.5]. A recent library has been proposed in [Zippo, Stea 2022].

2.2 Main Concepts of Network Calculus

Consider a flow f that crosses a system S (Figure 2.2a) and denote by w^{in} [resp., w^{out}] the observation point located at the input [resp., at the output] of S.

2.2.1 Cumulative Functions

Definition 2.4 (Cumulative function) *The cumulative function of f at the observation point w, noted $R_{f,w}$, is the function such that for $t \geq 0$, $R_{f,w}(t)$ is the number of bits of f that cross w during the time interval* $[0, t]$.

Throughout the thesis, we always assume that the time and the amount of data are continuous quantities and, up to Chapter 5, we assume that the time is universal (the same everywhere in the network). The origin of time 0 represents a time instant far away in the past, more specifically no source has sent any bit before $t = 0$. Figure 2.2b gives an example of two continuous cumulative functions $R_{f,w^{\text{in}}}$, $R_{f,w^{\text{out}}}$ for the situation of Figure 2.2a.

2.2.2 Backlog and Virtual Delay

If the system in Figure 2.2a is causal, then the amount of data that entered at t but did not leave is $x(t) = R_{f,w^{\text{in}}}(t) - R_{f,w^{\text{out}}}(t)$. If the system is also lossless, then $x(t)$ is the amount of data inside the system at t. Graphically, the backlog is the vertical distance between the

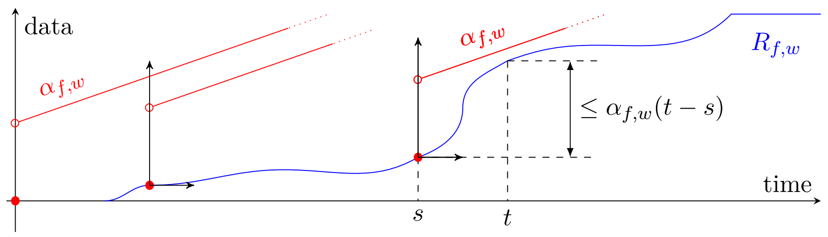


Figure 2.3: Graphical illustration of the arrival-curve constraint. The evolution of $R_{f,w}$ between s and t must remain below $\alpha_{f,w}(t-s)$, *i.e.*, below the arrival curve placed at $(s, R_{f,w}(s))$.

input cumulative function $R_{f,w^{\text{in}}}$ and the output cumulative function $R_{f,w^{\text{out}}}$ (Figure 2.2b). The horizontal distance $d(t)$ between the functions is the virtual delay. If the system serves the data of the flow in a first in, first out (FIFO) manner, then $d(t)$ is the delay that the bit arriving at t experiences in the system.

We emphasize that the above notions rely on three key assumptions, by decreasing order of importance: the causal assumption (the system does not produce or duplicate any data internally), the lossless assumption (the system does not lose any data), and the FIFO assumption. Most analyses of time-sensitive networks with network calculus rely on these three assumptions. In Chapter 4, we discuss a situation in which some of them are not valid.

In an asynchronous network, the sources generate the packets at random time instants, thus $R_{f,w^{\text{in}}}$ and $R_{f,w^{\text{out}}}$ are not known and cannot be used to compute the worst-case backlog or delay. Furthermore, we are usually not interested in the worst-case metrics of a particular physical network (e.g., in a particular car), but in the worst-case metrics across all the instances of a network specification (e.g., in all the cars that embark an instance of the network specification).

2.2.3 Arrival Curves

To circumvent the issue of not knowing the cumulative functions, we compute bounds on the worst-case performance metrics by replacing the cumulative arrival functions by the notion of arrival curves.

Definition 2.5 (Arrival Curve. Le Boudec, Thiran 2001, §1.2.1) *Consider a flow f and an observation point w. Denote by $R_{f,w}$ the cumulative function of f at the observation point w. $\alpha \in \mathfrak{F}$ is an arrival curve for f at observation point w if and only if*

$$\forall 0 \leq s \leq t, \quad R_{f,w}(t) - R_{f,w}(s) \leq \alpha(t-s)$$

We note such a function $\alpha_{f,w}$.

Strictly speaking, there exits a family of functions that all match Definition 2.5. To ease the notation, $\alpha_{f,w}$ is used in this thesis to describe a specific function within this family.

Theorem 2.1 (Combining Several Arrival Curves. Le Boudec, Thiran 2001, §1.2.3) *If α_1 and α_2 are arrival curves for f at w, so is their min-plus convolution $\alpha_1 \otimes \alpha_2$.*

Figure 2.3 illustrates the arrival-curve constraint: By "sliding" the starting point of the arrival curve on the cumulative function, the latter must always remain below the former.

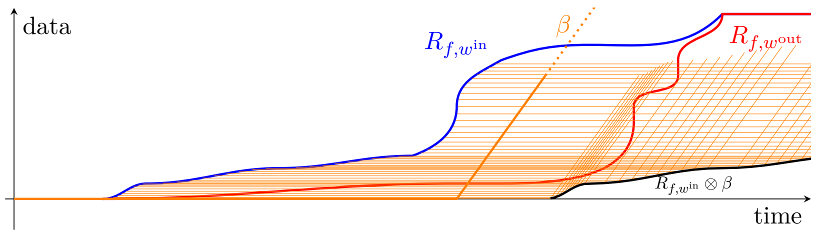


Figure 2.4: Graphical illustration of the service-curve constraint. We slide the starting point of the function β on the input cumulative function $R_{f,w^{\text{in}}}$. The lower bound of all the obtained values (the lower bound of the orange area) is the result $R_{f,w^{\text{in}}} \otimes \beta$ (in black). The output cumulative function $R_{f,w^{\text{ou}}}$ must remain above this black line.

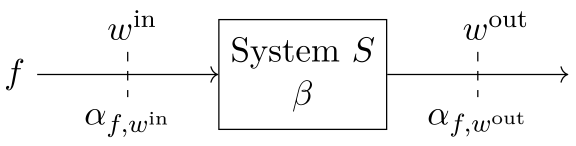


Figure 2.5: A flow f crosses a network element S. The observation point w^{in} [resp., w^{out}] is located at the input of the network element [resp., at its output]. The unknown cumulative functions are replaced by arrival-curve and service-curve constraints.

2.2.4 Service Curves

Definition 2.6 (Service Curve of a Network Element. Le Boudec, Thiran 2001, Def. 1.3.1) *We say that S offers to f the service curve $\beta \in \mathfrak{F}_0$ if $R_{f,w^{out}} \geq R_{f,w^{in}} \otimes \beta$.*

Graphically, the service-curve constraint is shown in Figure 2.4. By "sliding" the curve β (in orange) onto the function $R_{f,w^{\text{in}}}$ (in blue), we "paint" an orange area shown in Figure 2.4. The lower border of this zone (in black) gives the result of $R_{f,w^{\text{in}}} \otimes \beta$. Therefore, $\beta \in \mathfrak{F}_0$ is a service curve if and only if the output cumulative function $R_{f,w^{\text{out}}}$ is above this border.

2.3 The Main Results of the Network-Calculus Theory

In the previous section, the network-calculus concepts replace the unknown cumulative functions by lower- and upper-bounds. We can therefore replace Figure 2.2a by Figure 2.5, in which the flow is now modeled by an arrival curve $\alpha_{f,w^{\text{in}}}$ at the observation point w^{in} and the system is modeled by a service curve β that is offered to the flow f.

2.3.1 The Network-Calculus Three-Bound Theorem

Theorem 2.2 (The Three-Bound Theorem. Le Boudec, Thiran 2001, §1.4.1.) *Consider a causal system S and a flow f that crosses the system. Denote by w^{in} [resp., w^{out}] an observation point located at the input [resp., at the output] of the system (Figure 2.5). Assume that $\alpha_{f,w^{in}} \in \mathfrak{F}$ is an arrival curve for f at w^{in} and that the system S offers to f the service curve $\beta \in \mathfrak{F}_0$.*

- *The backlog of f inside S is upper-bounded*[a] *by $\mathfrak{v}(\alpha_{f,w^{in}}, \beta)$.*
- *If S is lossless, then the virtual delay $d(t)$ is upper bounded*[b] *by $\mathfrak{h}(\alpha_{f,w^{in}}, \beta)$. If S*

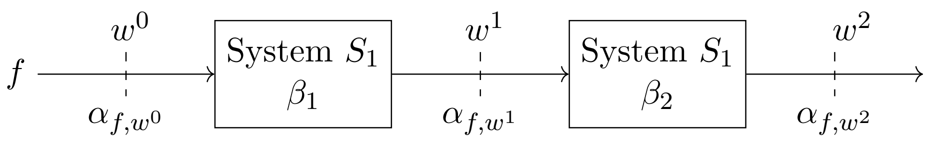


Figure 2.6: A flow f crosses two network elements S_1, S_2 in sequence.

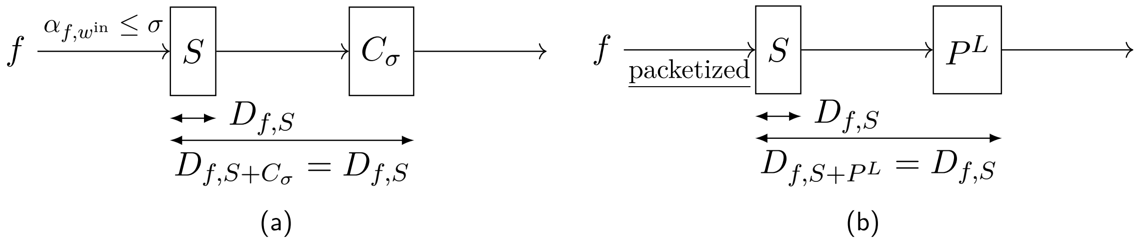


Figure 2.7: The "for-free" properties of the *greedy shaper* (a) and of the *packetizer* (b).

is lossless and FIFO, then the worst-case delay of f through S is upper bounded by $\mathfrak{h}(\alpha_{f,w^{in}}, \beta)$.

- *The curve $t \mapsto (\alpha_{f,w^{in}} \oslash \beta)(t)$ when $t > 0$, $t \mapsto 0$ otherwise, is an arrival curve for f at the output w^{out} that we can therefore denote by $\alpha_{f,w^{out}}$.*

[a] $\mathfrak{v}(\mathfrak{f}, \mathfrak{g}) = \sup_{t\geq 0}\{\mathfrak{f}(t) - \mathfrak{g}(t)\}$ is the vertical deviation between $\mathfrak{f}$ and $\mathfrak{g}$.
[b] $\mathfrak{h}(\mathfrak{f}, \mathfrak{g}) = \sup_{t\geq 0}\{\inf\{d; \forall d \geq 0 | \mathfrak{f}(t) \leq \mathfrak{g}(t+d)\}\}$ is the horizontal deviation between $\mathfrak{f}$ and $\mathfrak{g}$.

2.3.2 Concatenation of Systems

Theorem 2.3 (Concatenation of Systems. [Le Boudec, Thiran 2001], Thm. 1.4.6) *Consider a flow f that crosses two causal systems S_1 and S_2 in sequence (Figure 2.6). Assume that S_1 [resp., S_2] offers to f the service curve β_1 [resp., β_2]. Then, the concatenation of S_1 and S_2 offers to f the service curve $(\beta_1 \otimes \beta_2)$.*

Applying Theorem 2.3 then Theorem 2.2 in Figure 2.6 gives that $\mathfrak{h}(\alpha_{w^0}, \beta_1 \otimes \beta_2)$ is an upper bound on the delay of f through the concatenation of S_1 and S_2, which is always better than the successive application of Theorem 2.2 to S_1 then S_2 (*i.e.*, $\mathfrak{h}(\alpha_{f,w^0}, \beta_1) + \mathfrak{h}(\alpha_{f,w^1}, \beta_2) = \mathfrak{h}(\alpha_{f,w^0}, \beta_1) + \mathfrak{h}(\alpha_{f,w^0} \oslash \beta_1, \beta_2)$). This effect is known as *pay burst only once* (PBOO).

2.3.3 Greedy Shapers, Packetizers

Theorem 2.2 provides performance bounds for a flow through a system, as soon as the system can be modeled with a service curve. However, there exist also systems for which we can provide results that are stronger than those obtained from their service-curve representation.

Definition 2.7 (Greedy Shaper. Le Boudec, Thiran 2001, §1.5.1) *A greedy shaper with shaping curve σ, denoted C_σ is a causal, lossless, FIFO system that forces its output to be σ-constrained and releases the bits as soon as doing so does not violate the σ constraint.*

Theorem 2.4 (Properties of the Greedy Shaper. Le Boudec, Thiran 2001, §1.5)
– *If $\sigma \in \mathfrak{F}_0$ is sub-additive*[a]*, then $\mathcal{C}_\sigma$ is a system that provides σ as a service curve.*
– *If $\sigma \in \mathfrak{F}_0$ is sub-additive and if α is an arrival curve for f at the input of the greedy shaper, then $\alpha \otimes \sigma$ is an arrival curve for f at the output of the greedy shaper.*
– *Consider a flow f with arrival curve $\alpha_{f,w^{in}}$ that crosses a causal and FIFO system S and then a greedy shaper with shaping curve $\sigma \geq \alpha_{f,w^{in}}$ as in Figure 2.7a. Then the delay bound $D_{f,S}$ of f through S obtained with Theorem 2.2 is also a delay bound for f through the concatenation of S with the greedy shaper,* i.e., $D_{f,S+\mathcal{C}_\sigma} = D_{f,S}$.

[a] *i.e.,* $\forall s, t \geq 0$, $\sigma(t+s) \leq \sigma(t) + \sigma(s)$

Definition 2.8 (Packetizer. Le Boudec, Thiran 2001, §1.7.2) *A packetizer (denoted P^L) is a causal, lossless, FIFO system that transforms a fluid, bit-by-bit stream into a packetized stream by releasing the packet's bits only when the last one is received.*

Theorem 2.5 (Property of the Packetizer. Le Boudec, Thiran 2001, Thm. 1.7.1) *Consider a flow f that crosses a sequence of a system S followed by a packetizer P^L (Figure 2.7b). If f is packetized at the input of S, then the delay bound $D_{f,S}$ for f through S obtained with Theorem 2.2 is also a delay bound for f through the concatenation of S and P^L, which we denote as $D_{f,S+P^L} = D_{f,S}$ in Figure 2.7b.*

The packetizer increases, however, the burstiness of the flow f at its output thus can lead to increased delay bounds in downstream nodes. This burstiness increase is known to be bounded by $L_{\max}$ [Le Boudec, Thiran 2001, Thm. 1.7.1], the maximum packet size of the flow f. In Chapter 3, we provide a stronger result when the arrival rate of the bits at the input of the packetizer is constrained.

2.4 FIFO-per-Class Systems

In the previous sections, we considered a unique flow that crosses one or several systems. To apply Theorem 2.2, we assumed the knowledge of an arrival curve for f at the input of the system ($\alpha_{f,w^{\text{in}}}$) and the knowledge of a service curve β that the system *offers to* f. In a given time-sensitive network, f is unlikely to be the only flow that crosses S.

The time-sensitive networks studied in the thesis are FIFO-per-class networks: the system S segregates the flows into classes $\{\mathcal{C}_j\}_j$ and the flows within a same class are processed in a FIFO manner. In this thesis, we assume that the flows are statically assigned to the classes.

Consider a flow of interest and denote by $\mathcal{C}_i$ its class. Assume that we know an arrival curve $\alpha_{f,w^{\text{in}}}$ for f at the input of S and a service curve β_S offered by S to the entire aggregate of all the flows crossing S. To obtain performance guarantees for f, we need

- to understand which part $\beta_{S,\mathcal{C}_i}$ of the overall service β_S is specifically offered to the class $\mathcal{C}_i$. $\beta_{S,\mathcal{C}_i}$ is the residual service-curve for class $\mathcal{C}_i$. The answer depends on the system-level scheduling policy of the system, as discussed in Section 2.4.1.
- to understand what performance guarantees can be derived from $\beta_{S,\mathcal{C}_i}$ for f specifically. The answer depends on the FIFO policy (the class-level scheduling policy), the properties of which are discussed in Section 2.4.2.

2.4.1 Residual Service-Curves for Scheduling Policies Other than FIFO

Obtaining the residual service-curve $\beta_{S,\mathcal{C}_i}$ for $\mathcal{C}_i$ represents a consequential part of the network-calculus literature because many different scheduling policies exist. Additionally, the scheduling policies (as in the context of IEEE TSN) are rarely designed by network-calculus-aware researchers. Therefore, they sometimes lead to complex behaviors, the worst-case of which can be complex to compute. An excerpt of the literature for the most common scheduling policies is compiled in Table 2.1. Similar tables can be found in [Bouillard, Boyer, Le Corronc 2018, §7.4, §8.4].

Table 2.1: Overview of the Literature On the Computation of Residual Service-Curves

Scheduling Policy	References
Blind Multiplexing (also called Arbitrary Multiplexing)	[Le Boudec, Thiran 2001, §6.2.1] [Bouillard, Boyer, Le Corronc 2018, §7.2.1]
Preemptive Static Priority (P-SP)	[Bouillard, Boyer, Le Corronc 2018, §7.3.2]
Non-Premptive Static Priority (NP-SP) [IEEE 802.1Q, §8.6.8.1]	[Le Boudec, Thiran 2001, §6.2.1] [Bouillard, Boyer, Le Corronc 2018, §8.2.1]
Deficit Round-Robin (DRR) [Shreedhar, Varghese 1995]	[Boyer, Stea, Sofack 2012] [Bouillard 2021] [Tabatabaee, Le Boudec 2022]
Weighted Round-Robin (WRR) [Katevenis, Sidiropoulos, Courcoubetis 1991]	[Soni, *et al.* 2018] [Bouillard, Boyer, Le Corronc 2018, §8.2.4]
Interleaved Weighted Round-Robin (IWRR) [Katevenis, Sidiropoulos, Courcoubetis 1991]	[Tabatabaee, Le Boudec, Boyer 2021]
Credit-based shaper (CBQS) [IEEE 802.1Q, §8.6.8.2]	[Daigmorte, Boyer, Zhao 2018] [Mohammadpour, Stai, Boudec 2019] [Zhao, *et al.* 2018]
Time-Division Multiple Acess (TDMA) Also known as Time-Aware Shaper (TAS) [IEEE 802.1Qbv] [IEEE 802.1Q, §8.6.8.3-4, §8.6.9]	[Dang, Mifdaoui 2014] [Bouillard, Boyer, Le Corronc 2018, §8.2.5]

The blind-multiplexing policy (or arbitrary-multiplexing) is a conservative model that applies to any other scheduling policy. It has received much attention, especially for the *compositional approaches* (Section 2.5) because any delay bounds obtained under the blind-multiplexing policy is also valid under the FIFO policy.

2.4.2 The FIFO Scheduling Policy

Assume now the knowledge of $\beta_{S,\mathcal{C}_i}$, the service curve that the FIFO-per-class system S offers to the class-of-interest $\mathcal{C}_i$. Denote by $\alpha_{h,w^{\text{in}}}$ the arrival curve of each flow $h \in \mathcal{C}_i$ of the class at the input of S. We are interested in performance guarantees for a flow $f \in \mathcal{C}_i$.

The Aggregate Delay Bound is a Per-flow Delay Bound

A first approach to the FIFO policy is to consider the entire class aggregate as a unique flow. The sum of the individual arrival-curves $\sum_{h|h\in\mathcal{C}_i} \alpha_{h,w^{\text{in}}}$ is an arrival curve for the class aggregate at the input of S. Applying Theorem 2.2, we obtain that $D = \mathfrak{h}\left(\sum_{h\in\mathcal{C}_i} \alpha_{h,w^{\text{in}}}, \beta_{S,\mathcal{C}_i}\right)$ is a delay bound for the aggregate through the system S. By property of the FIFO policy, D is also a delay bound for any individual flow f, which describe the following residual service:

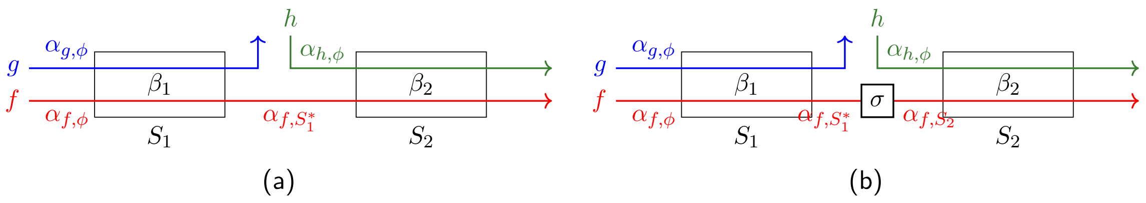


Figure 2.8: (a) Toy example for the introduction of the different compositional approaches. (b) Toy example with a greedy shaper (e.g., a transmission link) between the two systems.

Proposition 2.3 (FIFO residual service-curve with the total-flow analysis (TFA) Model. Bouillard, Boyer, Le Corronc 2018, Thm 7.4)
For each flow $f \in \mathcal{C}_i$, S offers to f the residual service-curve $\beta_f = \delta_{\mathfrak{h}(\sum_{h\in\mathcal{C}} \alpha_{h,w^{in}}, \beta_{S,\mathcal{C}_i})}$

In this thesis, we refer to Proposition 2.3 as the TFA model of the FIFO policy because this model is used in the total-flow analysis (TFA) method introduced later in Section 2.5.1.

A Family of Residual Service-Curves

The residual service-curve from Proposition 2.3 is not very good, and tighter results are available in the literature [Bouillard, Boyer, Le Corronc 2018, §7.3.1], [Le Boudec, Thiran 2001, §6.2.2]. However, contrary to the majority of the scheduling policies listed in Table 2.1, the FIFO scheduling policy is not well described by a unique service-curve, but by a family of residual service-curves, none of which dominate another:

Theorem 2.6 (FIFO Residual Service-Curves. Cruz 1998, Thm 4)
Denote by α_2 the curve $\alpha_2 = \sum_{h\in\mathcal{C},h\neq f} \alpha_{h,w^{in}}$. Then S offers to f the family of residual service curves: $\forall\theta \geq 0, \quad \beta_f^\theta : t \mapsto |\beta(t) - \alpha_2(t-\theta)|^+ \wedge (\delta_\theta(t))$

In this thesis, we refer to Theorem 2.6 as the SFA-model of the FIFO policy because this model is used in the separated flow analysis (SFA) method introduced later in Section 2.5.2.

2.5 Compositional Approaches for the Analysis of Feed-Forward FIFO-per-class Networks

Here, we give an overview of the challenges of computing end-to-end latency bounds for the flows in an entire FIFO-per-class network feed-forward network. To do so, we briefly describe four families of *compositional approaches* for computing end-to-end latency bounds in such networks. These methods compose with the different pieces of results and models presented in the previous sections to compute the end-to-end performance bounds in a full network, hence their name. The design of *compositional approaches* constitute an active field of research. As opposed to the single-server situation for which Theorem 2.2 is tight, computing tight end-to-end latency bounds in a FIFO-per-class network is indeed an NP-hard problem [Bouillard, Boyer, Le Corronc 2018, Thm. 10.2], [Bouillard, Jouhet, Thierry 2010].

We use the toy example shown in Figure 2.8a with two systems, S_1 and S_2 and three flows in the class of interest: f, g, h. We assume that S_1 offers to the class the service β_1 and S_2

offers to the class the service β_2. These service curves are obtained from Section 2.4.1. We denote by $\alpha_{f,\phi}, \alpha_{g,\phi}$ and $\alpha_{h,\phi}$ the arrival curve of the flows at their respective sources. We are interested in obtaining a latency bounds for f through the concatenation of S_1 and S_2.

2.5.1 Total Flow Analysis

The total-flow analysis (TFA) method [Schmitt, Zdarsky 2006, §3.2] relies on the TFA-model of the FIFO policy developed in Section 2.4.2. The method applies Proposition 2.3 to the systems in topological order.

Example: The aggregate arrival curve at the input of S_1 is $\alpha_{S_1} = \alpha_{f,\phi} + \alpha_{g,\phi}$. By Proposition 2.3, S_1 offers to f the residual service curve $\delta_{D_{f,S_1}}$ with $D_{f,S_1} = \mathfrak{h}(\alpha_{f,\phi} + \alpha_{g,\phi}, \beta_1)$ and $\alpha_{f,S_1^*} = \alpha_{f,\phi} \oslash \delta_{D_{f,S_1}}$ is an arrival curve for f at the output of S_1. Moving on to the next system and with the same arguments,

$$D_{f,S_2} = \mathfrak{h}(\alpha_{f,S_1^*} + \alpha_{h,\phi}, \beta_2) = \mathfrak{h}((\alpha_{f,\phi} \oslash \delta_{D_{f,S_1}}) + \alpha_{h,\phi}, \beta_2) \tag{2.1}$$

is a delay bound through S_2. Thus, $D_f = D_{f,S_1} + D_{f,S_2}$ is an end-to-end latency bound.

The TFA compositional approach is conceptually simple. It does not use the pay burst only once (PBOO) principle (*i.e.*, the concatenation property, Theorem 2.3), which, as discussed in Section 2.3.2, leads to loose delay bounds. However, TFA scales very well to large networks [Schmitt, Zdarsky 2006]. It provides the sufficient flexibility for modeling the line-shaping effect introduced in [Grieu 2004; Mifdaoui, Leydier 2017].

Example: Assume that a greedy shaper (Definition 2.7) is placed between S_1 and S_2 with shaping curve σ, as in Figure 2.8b. For example, a transmission link with capacity c is a greedy shaper with shaping curve $\sigma = \gamma_c$.

Then by application of Theorem 2.2, $\alpha_{f,S_2} = \gamma_{f,S_1^*} \otimes \sigma$ is an arrival curve for f at the output of the shaper, *i.e.*, at the input of S_2. Clearly, $\gamma_{f,S_1^*} \otimes \sigma \leq \gamma_{f,S_1^*}$, thus the delay bound D_{f,S_2} in S_2 computed with the greedy shaper is less than the delay bound obtained with (2.1). Considering the transmission links as greedy shapers has a beneficial effect on the end-to-end latency bounds; this is known as the line-shaping effect effect. This effect is used in Chapters 3 and 6.

As we discuss in Chapters 3 and 6, the flexibility of TFA enables the modeling of a wide variety of systems and phenomenons. For this reason, we select TFA for implementing the theoretical contributions of this thesis.

2.5.2 Separated Flow Analysis

The separated flow analysis (SFA) methods [Schmitt, Zdarsky 2006, §3.3] aim to apply the PBOO principle by concatenating the residual service-curves obtained with Theorem 2.6. They operate on a per-flow basis.

Example: As per Theorem 2.6, S_1 offers to f the family of residual service-curves

$$\left\{\beta_{1,f}^{\theta_1} : t :\mapsto |\beta_1(t) - \alpha_{g,\phi}(t-\theta_1)|^+ \wedge \delta_{\theta_1}(t); \quad \forall \theta_1 \geq 0\right\} \tag{2.2}$$

and S_2 offers to f the family of residual service-curves

$$\left\{\beta_{2,f}^{\theta_2} : t :\mapsto |\beta_2(t) - \alpha_{h,\phi}(t-\theta_2)|^+ \wedge \delta_{\theta_2}(t); \quad \forall \theta_2 \geq 0\right\} \tag{2.3}$$

Hence the concatenation of S_1 and S_2 offers to f the family of service curves

$$\left\{\beta_{1,f}^{\theta_1} \otimes \beta_{2,f}^{\theta_2}; \quad \forall \theta_1 \geq 0, \forall \theta_2 \geq 0\right\} \tag{2.4}$$

and for any $\theta_1, \theta_2 \geq 0$, the horizontal deviation $\mathfrak{h}(\alpha_{f,\phi}, \beta_{1,f}^{\theta_1} \otimes \beta_{2,f}^{\theta_2})$ is a delay bound for f through the concatenation of S_1 and S_2, thus $\min_{\theta_1 \geq 0, \theta_2 \geq 0} \mathfrak{h}(\alpha_{f,\phi}, \beta_{1,f}^{\theta_1} \otimes \beta_{2,f}^{\theta_2})$ gives a delay bound for f through S_1 and S_2.

In the case of FIFO multiplexing, the SFA approaches typically require finding the optimal values for the θ parameters of Theorem 2.6 applied to each FIFO system. This number of θ parameters increases with the length of the flow path, thus increasing the complexity of SFA.

In [Bouillard, Boyer, Le Corronc 2018, §10.4.3], Bouillard, Boyer and Le Corronc indicate that there exists also a family of intermediate methods in which the flows are aggregated into groups. This is also similar to [Grieu 2004, §3.2.3.1].

Also, note that SFA does not compute α_{f,S_1^*}, the arrival curve of f at the output of S_1. Hence, considering a greedy shaper between S_1 and S_2 does not improve the delay bounds. Indeed we know from Section 2.3.3 that a greedy shaper is more powerful than its service-curve representation. In Figure 2.8b, the end-to-end service is $\beta_{1,f}^{\theta_1} \otimes \gamma_c \otimes \beta_{2,f}^{\theta_2}$, which is simply equal to $\beta_{1,f}^{\theta_1} \otimes \beta_{2,f}^{\theta_2}$ if c is large enough (Proposition 2.1).

2.5.3 Tandems and Pay Multiplexing Only Once

Tandems networks are a class of networks in which the research of compositional approaches has been very active. A tandem network is a network in which the edges can be placed on a line and the path of each flow is a continuous sub-path of this line. Among the tandems, the *nested* tandems are such that an order $f_1, f_2, f_3, \ldots$ of the flows can be performed such that for any i, the path of f_i is nested in the path of f_{i+1}. The analysis of tandems networks can exploit a phenomenon known as the pay multiplexing only once (PMOO) that extends the pay burst only once (PBOO) principle [Fidler 2003].

Example: The example in Figure 2.8 is a tandem. If we remove flow g, it is a nested tandem (h is nested in f).

In [Lenzini, *et al.* 2006], the Least Upper Delay Bound (LUDB) approach relies on PMOO and is applied in FIFO sink-tree networks (in which all flows leave the network at the same server). Lenzini *et. al.* prove that for FIFO sink-tree networks, their bound is tight. Sink-tree networks have been considered of particular interest for sensor networks [Schmitt, Roedig 2005; Schmitt, Zdarsky, Roedig 2006; Roedig, Gollan, Schmitt 2007].

In [Lenzini, Mingozzi, Stea 2007], the LUDB approach is extended to nested tandems, but it loses its tightness [Bouillard, Boyer, Le Corronc 2018, §10.3.1]. A software tool that implements the results of the above papers was designed in [Bisti, *et al.* 2010]. In [Lenzini, Mingozzi, Stea 2008; Bisti, *et al.* 2008] a methodology for general tandems is proposed in which the tandem is cut into nested sub-tandems. For any given tandem, several ways of cutting it into sub-tandems exist. Hence, an important challenge with the LUDB approach, and with the PMOO principle in general, is finding an optimal transformation of a tandem into nested sub-tandems.

In [Schmitt, Zdarsky 2006] and [Schmitt, Zdarsky, Martinovic 2008] the authors focus on the arbitrary-multiplexing policy. As mentioned in Section 2.4.1, the arbitrary-multiplexing policy does not make any assumptions about the scheduling policy, thus the delay bounds obtained with this policy are also valid for FIFO networks. Here, an important difference is that the application of the PMOO approach for arbitrary multiplexing is not limited to nested tandems but is also possible for general tandems. Another key difference is that the blind-multiplexing PMOO approach does not always perform better than the blind-multiplexing SFA approach [Schmitt, Zdarsky, Fidler 2008]. The PMOO method can be extended to feed-forward networks by selecting a flow of interest, considering its path as the tandem and computing the arrival curve of the interfering flows when they enter this tandem. An important challenge here is to tightly bound the traffic of the interfering flows [Bondorf, Schmitt 2016b; Bondorf, Schmitt 2016a].

In [Bouillard, Nowak 2015], the authors computes the exact worst-case delay in general tandems with arbitrary multiplexing. In [Bondorf, Nikolaus, Schmitt 2017], Bondorf, Nikolaus and Schmitt rely on the SFA, LUDB and arbitrary-multiplexing PMOO to design an algorithm that performs an exhaustive yet efficient search of the optimal tandem decomposition, thus resulting in one of the best algorithms in the trade-off between scalability and bound tightness. Later in [Geyer, Bondorf 2019], the tandem matching algorithm in [Bondorf, Nikolaus, Schmitt 2017] is replaced by a deep-learning approach, thus resulting in an even faster execution time.

A recent and totally different approach for transforming a general tandem into a (unique) nested tandem is the flow prolongation approach [Bondorf 2017] in which the path of the flows in the tandem are prolonged. Here again, several flow prolongations per tandem can be envisioned and, in [Geyer, Scheffler, Bondorf 2022], the best flow prolongations are predicted by a machine-learning algorithm.

The TFA, SFA and PMOO are often combined with a min-plus back-end and integrated in software tools, such as in the DISCO network calculator [Schmitt, Zdarsky 2006; Gollan, *et al.* 2008; Bondorf, Schmitt 2014], WoPaNets [Mifdaoui, Ayed 2010], Pegase [Boyer, *et al.* 2010], etc.

2.5.4 Linear Programming

We finally note a radically different approach that does not directly rely on the network calculus results: The linear-programming approach, introduced in [Bouillard, Stea 2012; Bouillard, Stea 2015], computes the worst-case delay bound as the solution of a linear optimization-problem, where the arrival-curve, service-curve and FIFO constraints are replaced by linear constraints in the linear program. However, the number of variables is exponential with the size of the network, hence LP is known to be applicable for networks of only a dozen nodes.

In [Bouillard 2022], Bouillard introduces the polynomial-size linear program (PLP) in which a heuristic removes some of the decision variables of the LP approach to keep the size of the problem polynomial within the size of the network. Interestingly, the PLP approach can easily capture the line-shaping effect [Bouillard 2022, §4.1] and can even be extended to networks with cyclic dependencies, which we further discuss in Section 3.1.4.

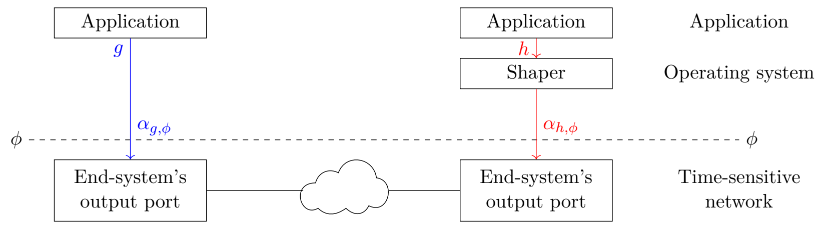


Figure 2.9: Illustration of the boundary ϕ of the network. For each flow f entering the network, we assume the knowledge of an arrival curve $\alpha_{f,\phi}$ for f at the input boundary ϕ at its source. $\alpha_{f,\phi}$ is either obtained from the specification of the application (case of g), or enforced by the operating system (case of h).

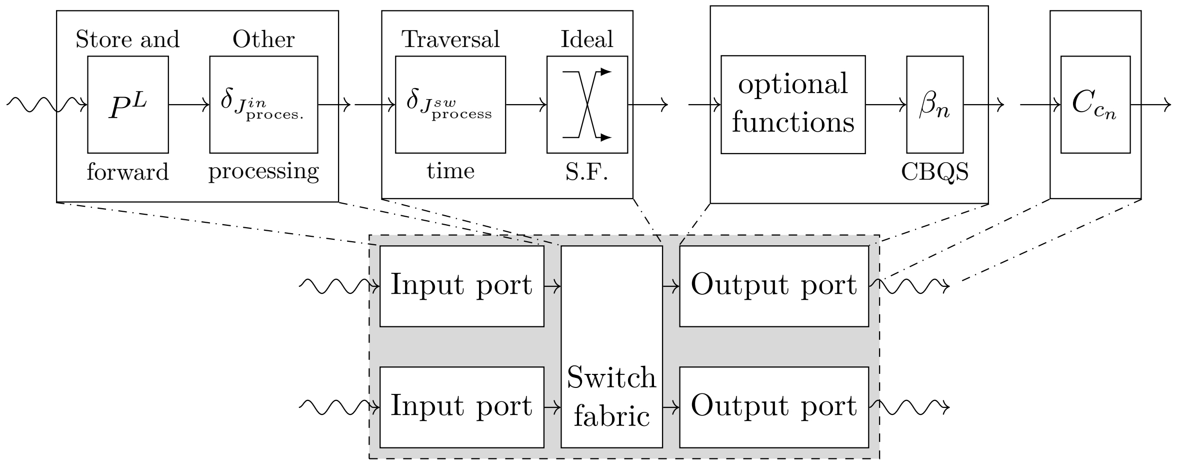


Figure 2.10: Model for any device of the time-sensitive network. Each device is made of input ports, output ports and a switching fabric. The input port contains a store-and-forward step and bounded technological latencies. The switching fabric contains an ideal switching fabric and bounded technological latencies. The output port contains a set of optional functions and a FIFO class-based queing subsystem (CBQS). The transmission links act as greedy shapers.

2.6 Network-Calculus Model For Time-Sensitive Networks

In this section, we build upon the related work on modeling time-sensitive networks with network calculus to introduce the network-calculus model that we use throughout the thesis for obtaining end-to-end performance guarantees when focusing on a class of interest.

2.6.1 The Sources

We denote by ϕ the boundary of the time-sensitive network, as illustrated in Figure 2.9. For any flow f that enters the network, we assume the knowledge of an input arrival-curve for f at the boundary ϕ, denoted $\alpha_{f,\phi}$. In time-sensitive networks, $\alpha_{f,\phi}$ is either obtained from the source specifications or is enforced by a shaper in the operating system [AFDX]. $\alpha_{f,\phi}$ can also be obtained from the TSN and DetNet specifications [48], [Zhao, *et al.* 2018; Maile, Hielscher, German 2020].

In the following, we consider a time-sensitive network device (Figure 2.10), it can either be an IEEE TSN *bridge* or an IETF DetNet *router*. The device is made of *input ports*, *output ports* and a *switching fabric*. Devices are connected together through transmission links that we assume to have fixed capacity.

2.6.2 The Input Port

We assume that each *input port* contains a *store-and-forward step* and *other processing steps.* We model the store-and-forward step by using the network-calculus packetizer (Definition 2.8). We assume that the *other processing steps* (decryption, cyclic redundancy check, etc.) are performed in a causal, lossless, and FIFO manner and that their duration is bounded in $[d^{in}_{\text{proces.}}, D^{in}_{\text{proces.}}]$. As such, we model them as causal, lossless, and FIFO system that provides the service curve $\delta_{J^{in}_{\text{proces.}}}$ (Figure 2.10), with $J^{in}_{\text{proces.}} \triangleq D^{in}_{\text{proces.}} - d^{in}_{\text{proces.}}$ and δ_J is the bounded-delay curve (Figure 2.1c).

2.6.3 The Switching Fabric

The switching fabric connects the inputs ports to the output ports. For a given flow, the switching fabric forwards the flow to one or several output ports, depending on the path of the flow. When a packet of the flow reaches the switching fabric, we say that the latter copies the content of the packet, *i.e.*, the data unit, into one or several new packets that are forwarded to the corresponding output ports.

Because of this copy process, the switching fabric is not assumed to be globally causal. However, for any input connection of the switching fabric and for any output connection, we assume that the pair (input,output) is causal, lossless, FIFO. The time taken by a data unit to cross this pair is assumed to be bounded between known values $[d^{sw}_{\text{proces.}}, D^{sw}_{\text{proces.}}]$.

Therefore, we model the switching fabric as a causal, lossless, FIFO system that provides the service curve $\delta_{J^{sw}_{\text{proces.}}}$, followed by an *ideal switching fabric* (the box with the arrows in Figure 2.10), of which any pair (input,output) is causal, lossless FIFO and where $J^{sw}_{\text{proces.}} = D^{sw}_{\text{proces.}} - d^{sw}_{\text{proces.}}$.

2.6.4 The Output Port

An overview of the output-port forwarding process in an IEEE TSN *bridge* is provided in Figure 2.11, based on the standards [IEEE 802.1Q; IEEE 802.1Qcr; IEEE 802.1CB].

The Optional Functions

The packets coming from the switching fabric are first distributed by a classification step to a set of optional processing steps (or directly to the queue of the class). Each optional processing step can process either one flow, the entire class aggregate, or any subset of flows from the class of interest. In general, this part of the forwarding process is neither lossless nor FIFO for the class.

In this thesis, we refer to this part simply as the “optional functions” for the class of interest in this output port (dashed-blue box in Figure 2.11). Three main optional functions are studied in this thesis:

- the traffic regulators such as the per-flow regulator (PFR) or the interleaved regulator (IR). They are implemented in TSN under the name *asynchronous traffic shaping* (ATS)

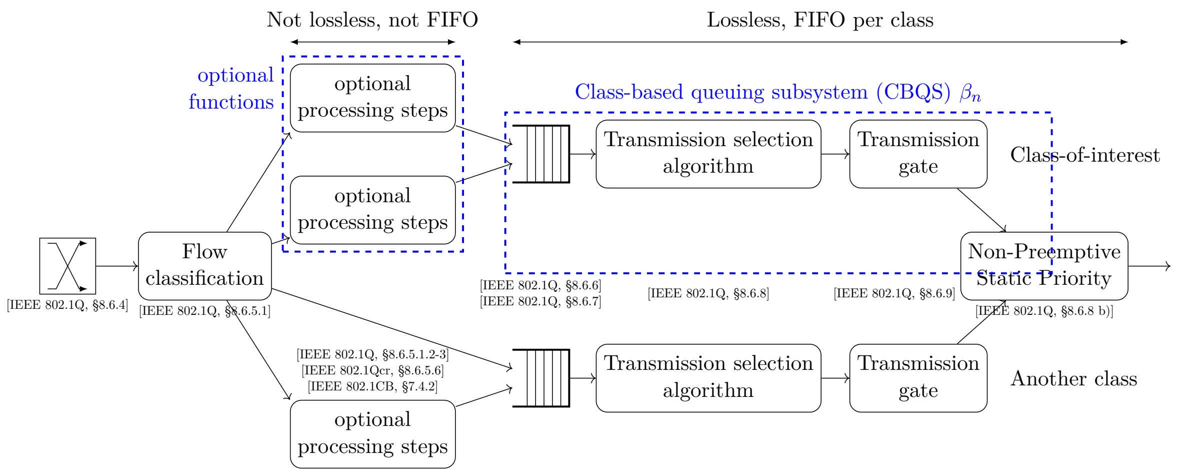


Figure 2.11: Overview of the forwarding process in an IEEE TSN output port n and mapping with the model used in this thesis (in dashed blue). We separate the content of the output port into the *optional functions* and the class-based queing subsystem (CBQS).

[IEEE 802.1Qcr] and in DetNet under the name *shapers* [RFC 2475, §2.3.3.3]. They are widely studied in the literature [Wandeler, Maxiaguine, Thiele 2006; Specht, Samii 2016; Le Boudec 2018; Mohammadpour, *et al.* 2018]. The main properties of the regulators are provided in Chapter 3 and are analyzed in various situations throughout the thesis.

- the packet-elimination function (PEF). This function is implemented in TSN under the name *frame replication and elimination for redundancy* (FRER) [IEEE 802.1CB] and in DetNet under the name *packet-elimination function* (PEF) [RFC 8655, §3.2.2.2.]. This function is studied in Chapter 4.
- the packet-ordering function (POF). This function is implemented in DetNet under the same name [RFC 8655, §3.2.2.2.], but it has no equivalent in TSN. The POF has been analyzed in [Mohammadpour, Le Boudec 2021] and its joint use with PEF is studied in Chapter 4.

The specific models for the above functions are deferred to the relevant chapters.

The Class-Based Queuing Subsystem

The combination of the forwarding steps located after the class buffer (transmission selection algorithm, transmission gates) determines the service that the class receives.

The Transmission Selection Algorithms (TSAs) are asynchronous scheduling mechanisms. In general, they can be mapped to the scheduling policies mentioned in Table 2.1. [Maile, Hielscher, German 2020] gives an overview of the TSAs for which a network-calculus service-curve representation has been obtained in the literature. For example, the *strict priority* TSA [IEEE 802.1Q, §8.6.8.1] emulates the non-preemptive static priority scheduling policy of Table 2.1 thus can be modeled as a service-curve element. The *credit-based shaper* TSA [IEEE 802.1Q, §8.6.8.2] maps to the credit-based shaper scheduling policy of Table 2.1. Its service-curve representation is analyzed in [Mohammadpour, Stai, Boudec 2019; Zhao, *et al.* 2021; Zhao, Pop, Steinhorst 2021]. The *enhanced transmission selection* TSA [IEEE

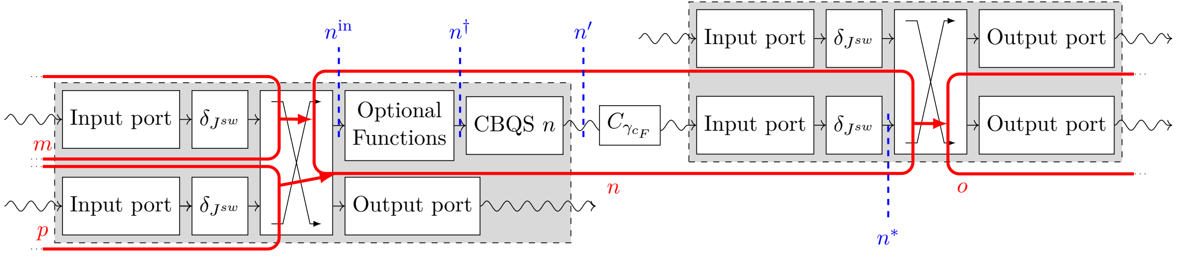


Figure 2.12: Relation between the bridges (in gray boxes), the vertices of the underlying graph (in thick red ovals) and the observation points (dashed blue lines). The vertex n models the output port n, together with the transmission link, the input port on the remote device and the latency of the switching fabric in the remote device. Due to the one-to-one mapping between output ports and vertices, the notation n describes equivalently the vertex n, the ouput port n or the CBQS n.

802.1Q, §8.6.8.3] can be configured as a deficit round-robin scheduling policy that is analyzed in [Boyer, Stea, Sofack 2012; Tabatabaee, Le Boudec 2022].

The configuration of the transmission gates results in a synchronous scheduling mechanism known as the time-aware shaper (TAS). The configuration of the time-aware shaper is far more complex than the configurations of the TSAs because the open/closed cycles for the gates must be computed. The service-curve modeling of the time-aware shaper is analyzed in [Zhao, Pop, Craciunas 2018] and in combination with TSAs in [Zhao, *et al.* 2018; Daigmorte 2019; Daigmorte, Boyer, Zhao 2018; Zhao, Pop, Steinhorst 2021; Zhao, *et al.* 2021]. The *cyclic queuing and forwarding* [IEEE 802.1Qch] was introduced as an attempt to ease the configuration of the gates.

Choosing the best set of algorithms parameters for a specific network remains an open question [NAVET, MAI, MIGGE 2019; Samson, *et al.* 2021], and several performance comparison relying on network calculus exist [Nasrallah, *et al.* 2019; Zhao, Pop, Steinhorst 2021].

In the thesis, we assume simply that the combination of the transmission-selection algorithm, the transmission gates and the non-preemptive static priority in output port n can be modeled for the class of interest as a unique causal, lossless, and FIFO system that offers to the class the service curve β_n, where β_n can be obtained, for example, from the literature overview provided in [Maile, Hielscher, German 2020; Zhao, Pop, Steinhorst 2021].

2.6.5 The Transmission Links

For each output port n, we assume that the transmission link connected at the output of n has a fixed capacity c_n and a fixed propagation time $T_n > 0$. We note $T_{\min} = \min_n T_n$ the minimum of the propagation time for all transmissions links in the network. We model each link as a greedy shaper (Definition 2.7) $C_{\gamma_{c_n,0}}$ of shaping curve $\gamma_{c_n,0}$ (Figure 2.1a).

2.6.6 The Graph Induced By Flows, Flow Graph

Definition 2.9 (Graph Induced by Flows, Flow Graph) *For a network and a class of interest, the graph induced by flows (GIF) is the directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where*

- *For each output port n in the network, $\mathcal{V}$ contains a vertex with identical name (n) that includes the output port, the remote input port and the variable-delay element* *of the switching fabric at the remote device (see Figure 2.12).*

- *For each n, m two vertices of $\mathcal{V}$, there exists a directed edge (n, m) in $\mathcal{E}$, if and only if at least one flow of the class of interest crosses the output port n and, immediately after, the output port m .*

For f a flow in the class of interest, the graph of flow f, $\mathcal{G}(f)$ is the sub-graph of $\mathcal{G}$ obtained by keeping only the vertices of which the output port is crossed by f. We note $f \ni n$ to denote the fact that f crosses the output port n, i.e., *n is a vertex of $\mathcal{G}(f)$. We note $f \ni (n, m)$ to denote that f crosses m immediately after n,* i.e., *(n, m) is a directed edge in $\mathcal{G}(f)$.*

For the moment, we assume that $\mathcal{G}(f)$ is a multi-cast tree with one or several destinations. Later in Chapter 4, we relax this assumption to allow for redundant paths.

2.6.7 Observation Points

For a vertex n, we define the observation points n^{in}, $n^{\dagger}$, n' and n^{*} as in Figure 2.12.

Conclusion

In this chapter, we have introduced the network-calculus framework for computing latency bounds in time-sensitive networks. We first gave an overview of the theoretical grounds of the theory, with the min-plus algebra in Section 2.1 and the main concepts of arrival curves and service curves in Section 2.2. Then in Section 2.3 we have provided the Three-Bound Theorem (Theorem 2.2) that is central in the theory for computing performance bounds (latency, backlog, output burstiness).

Afterwards, we have provided an overview of several active fields of research. We have discussed in particular the question of the residual service-curve computation in Section 2.4 as well as the properties of the FIFO policy. We have also provided an overview of the different *compositional approaches* for computing end-to-end latency bounds in FIFO-per-class networks. For its simplicity and flexibility, we select the TFA approach to implement our theoretical contributions.

Last we have provided the overall network-calculus model for the devices and the flows in time-sensitive networks. This model is used in Chapter 3, Chapter 4, Section 5.5 of Chapter 5 and Chapter 6. A summary of the model with a list of notations is available in Appendix A. The model is based on the TSN standards and on the related work on the analysis of TSN mechanisms using network-calculus. We have noted that the *bounded-latency* mechanisms (*schedulers*, *shapers*, etc.) are studied in numerous occasions with network calculus in the literature, as outlined in Figure 1 of this thesis' Introduction.

In the second part of the thesis, we use the network-calculus framework and the model described in this chapter to analyze the effects on the latency bounds of the new mechanisms identified in Figure 1. In the next chapter, we first analyze the effects of multi-path topologies on the performance bounds of time-sensitive networks.

Part II

Theoretical Contributions

Chapter 3

Cyclic Dependencies

"If one other Solarian exists to dispute my absolute mastery over any part of my land, over any robot or living thing or object, my freedom is limited. Since other Solarians exist, the limitation on freedom must be removed as far as possible by separating them all to the point where contact is virtually nonexistent."

Sarton Bander

Isaac Asimov, *Foundation and Earth.*

Contents

3.1 Related Work 38
- 3.1.1 Source of Cyclic Dependencies in Time-Sensitive Networks 38
- 3.1.2 The Issue of Cyclic Dependencies in Time-Sensitive Networks 39
- 3.1.3 Preventing the Cyclic Dependencies 39
- 3.1.4 Solving the Cyclic Dependencies 40
- 3.1.5 Using Traffic Regulators to Break the Cyclic Dependencies 42
- 3.1.6 Contributions in This Chapter 44

3.2 LCAN: an Optimal Algorithm for Breaking All Cyclic Dependencies 45
- 3.2.1 Formulation of the Problem with PFRs 46
- 3.2.2 Formulation of the Problem with IRs 46
- 3.2.3 Solving the MFAS Problem 50

3.3 FP-TFA: Computing Tight Delay Bounds in Networks with Cyclic Dependencies 50
- 3.3.1 A Novel Result on the Effect of Packetizers 51
- 3.3.2 Delay Bound within a Vertex 51
- 3.3.3 Extension of the FP-TFA Approach to Networks with Cyclic Dependencies 55
- 3.3.4 Extension of the FP-TFA Approach to Networks With Regulators 57

3.4 Evaluation of the Partial-Deployment Approach 58
- 3.4.1 Network Description 58
- 3.4.2 Parameters of Interest 58
- 3.4.3 Effect of the Network Size on the Performance of low-cost acyclic network (LCAN) 59
- 3.4.4 Effect of the Network Load on the Latency Bounds 59
- 3.4.5 Effect of the Line Shaping 60

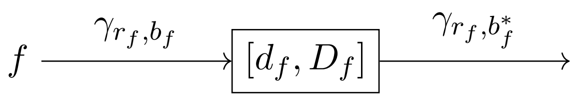


Figure 3.1: Phenomenon of burstiness increase: The burstiness of the flow f increases through the network element due to the jitter that it suffers within the network element.

As we have discussed in Chapter 1, Section 1.3, time-sensitive networks not only support general topologies but foster the use of multi-path topologies, because the latter include alternative paths that reduce the reconfiguration effort.

When the flows are mapped, however, on theses multi-path topologies, their paths can induce cyclic dependencies: the graph induced by flows (GIF) can contain cycles. In the last chapter, we presented the state of the art of computing delay bounds in FIFO-per-class feed-forward networks, *i.e.*, networks with no cyclic dependencies. Much less related work is available for networks with cyclic dependencies.

Indeed, cyclic dependencies are a well-known issue in time-sensitive networks because they can lead to unbounded delays [Andrews 2009]. Consequently, obtaining the proof of determinism for the networks with cyclic dependencies is a much more challenging issue than for the feed-forward networks (networks without cyclic dependencies).

Therefore, our focus of this chapter is the following: guaranteeing deterministic worst-case latency bounds in time-sensitive networks, even if they have cyclic dependencies, while minimizing the deployment costs and keeping high-scalability architectures.

In Section 3.1, we first discuss the issues caused by cyclic dependencies and the way they are handled in the literature. We then introduce our new approach based on the partial deployment of traffic regulators. In Section 3.2, we present low-cost acyclic network (LCAN), an algorithm for identifying the optimal positions of the regulators. In Section 3.3, we present FP-TFA, an algorithm based on the total-flow analysis (TFA) approach for computing latency bounds in networks with cyclic dependencies and/or regulators. Finally, we evaluate the performance of our partial-deployment approach on a parametric grid network, in Section 3.4.

Part of the material presented in this chapter was published in [Thomas, Le Boudec, Mifdaoui 2019]. Throughout the whole chapter we assume that each flow f is constrained at the source by a leaky-bucket arrival curve $\alpha_{f,\phi} = \gamma_{r_f,b_{f,\phi}}$ with rate r_f and burst at source $b_{f,\phi}$: over any window of duration t, the source of the flow cannot send more than $r_f t + b_{f,\phi}$ bits.

3.1 Related Work

3.1.1 Source of Cyclic Dependencies in Time-Sensitive Networks

When such a flow f crosses a server (Figure 3.1), the jitter that it suffers in the server (difference between its worst-case delay D_f and its best-case delay d_f) increases its burstiness $b_f^* > b_f$. This burstiness is then propagated to the next server. In some networks, this burst propagation can lead to situations in which the burstinesses of the flows have a cyclic dependency on each other.

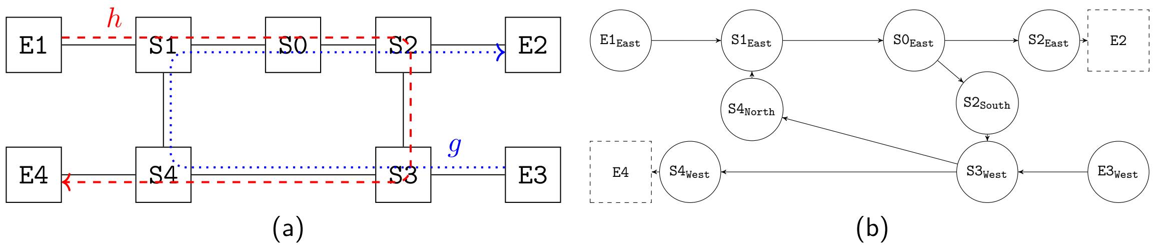


Figure 3.2: Toy example of a network with a cyclic dependency. (a) The physical topology and the paths of the two flows. (b) The graph induced by flows (GIF). Each vertex represents an output port. The destination end systems E2 and E4 are shown for convenience, but only output ports are present as vertices in the actual graph.

Example: This is the case for the FIFO-per-class toy example shown in Figure 3.2a, where boxes S1 to S4 are switches, E1 to E4 are end systems, and h and g are leaky-bucket constrained flows of the same class. When h enters S1, it competes with flow g to exit S1 in the direction of S2. This interference generates a jitter whose worst case depends on the burstiness of both flows h and g before S1. The jitter causes an increase of the worst-case burstiness of h and g after S1. Focusing on h, this new burstiness is propagated to S2, where h suffers again a jitter that further increases its burstiness. At S3, the propagated burstiness of h competes with the fresh flow g, thus creating a burstiness increase for h and g. And the burstiness of g continues to increase between S3 and S1. Therefore, we reach a cyclic dependency because the burstiness of g at the input of S1 depends on the jitter that h suffers within S1 that, in turn, depends on the burstiness of g at the input of S1.

To explicit the cyclic dependency, we can build the *graph induced by flows* (GIF) (Definition 2.9). Let us use the cardinal directions to distinguish the different output ports for a given switch of Figure 3.2a. Then, the GIF for the toy example is shown in Figure 3.2b. For example, $(\mathtt{S3}_{\mathtt{West}}, \mathtt{S4}_{\mathtt{North}})$ is a directed edge of the graph because g crosses $\mathtt{S4}_{\mathtt{North}}$ immediately after $\mathtt{S3}_{\mathtt{West}}$, in Figure 3.2a. We note that the graph contains a cycle that characterizes the cyclic dependency: the network is not feed-forward.

3.1.2 The Issue of Cyclic Dependencies in Time-Sensitive Networks

Obtaining performance guarantees on non-feed-forward networks is a challenging issue much more than for feed-forward networks. This is because the FIFO scheduling policy (used in FIFO-per-class networks) belongs to the set of aggregate scheduling policies that are not universally stable: There exists FIFO-per-class networks that are under-loaded (the utilization factor of each node is below 1) but in which the delays and the backlogs are unbounded [Andrews 2009].

3.1.3 Preventing the Cyclic Dependencies

If cyclic dependencies are anticipated in a network under design, then the designer could choose to avoid them in the first place. Many industrial networks continue to avoid the cyclic dependencies and to guarantee a feed-forward network by using specific routing constraints computed prior to the flow paths [AFDX]. This approach benefits from the research on deadlock prohibition in wormhole-routing networks [Li, Gu 2009]. A wide variety of algorithms have been proposed. For example, the X-Y routing [Li, Gu 2009] is applicable for grid networks, the turn-prohibition algorithm is applicable for general topologies [Starobinski, Karpovsky, Zakrevski 2003]. An overview of the different approaches can be found in [Finzi, Craciunas 2019, §3.1].

Example: The cycle in the toy example can be avoided by preventing only one turn: for example, the turn $(\mathtt{S0}, \mathtt{S2}, \mathtt{S3})$ in Figure 3.2a. This means that a new route for the flow h must be computed: for example, the shortest path $\mathtt{E1} \to \mathtt{S1} \to \mathtt{S4} \to \mathtt{E4}$.

We note that many of the algorithms can be written as different solvers of the same minimum feedback arc set (MFAS) problem, a well-known NP-complete problem of graph theory [Baharev, Schichl, Neumaier 2015] in which a directed graph (the GIF) is made acyclic by removing as few of its edges as possible. Each removed edge corresponds to a prohibited turn and the goal is to remove all cyclic dependencies while minimizing the turns that are prohibited. For example, the turn-prohibition algorithm [Starobinski, Karpovsky, Zakrevski 2003] can be written as a sub-optimal solver for the MFAS problem [Fidler, Einhoff 2004].

Example: In the toy example, we note that prohibiting turn $(\mathtt{S0}, \mathtt{S2}, \mathtt{S3})$ is equivalent to prohibiting the edge $(\mathtt{S0}_{\mathtt{East}}, \mathtt{S2}_{\mathtt{South}})$ from the GIF (Figure 3.2b). This edge is an optimal solution to the MFAS problem for the graph of Figure 3.2b.

Routing constraints ensure feed-forward networks that can be analyzed using the feed-forward techniques described in Chapter 2, Section 2.5. But they give little latitude to the network architect on the mapping of the flows. They face configuration issues in the context of large-scale networks that are at the core of IEEE TSN and IETF Detnet working groups. Finally, they can be incompatible with redundancy requirements. Consequently, we do not consider this class of solution in the following parts of this chapter.

Example: If both communications $\mathtt{E1} \to \mathtt{E4}$ and $\mathtt{E3} \to \mathtt{E2}$ require two paths each, then the only possible redundant paths are those used by h and g in Figure 3.2a, and a cyclic dependency necessarily exists. As a result, no routing algorithm can find two alternative paths for each communication, without creating a cyclic dependency.

3.1.4 Solving the Cyclic Dependencies

When the paths of the flows cannot be changed, we can use one of the methods for obtaining latency bounds with network calculus on networks with cyclic dependencies. With each of these methods, there exists a value, $u_{\text{crit}} \in [0, 1]$ that represents the supremum of the values of the network load at which the considered method can compute end-to-end delay bounds for each flow: When the network load u is such that $u < u_{\text{crit}}$, then a delay bound can be obtained for each flow using the method. When $u > u_{\text{crit}}$, the considered method cannot conclude on the stability of the network.

The value of the critical load with feed-forward networks is always $u_{\text{crit}} = 1$: a delay bound can always be obtained with network calculus for each flow of an under-loaded feed-forward network, by using one of the methods presented in Chapter 2. Whereas, for the methods that are tailored to networks with cyclic dependencies, we often have $u_{\text{crit}} < 1$.

In [Charny, Le Boudec 2000], Charny and Le Boudec provide a first value for u_{crit} for FIFO-per-class networks: If the flows of the network traverse each at most k hops, then $u_{\text{crit}} = \frac{1}{k-1}$ with [Charny, Le Boudec 2000].

Example: Each of h and g crosses five hops. The result from [Charny, Le Boudec 2000] gives $u_{\text{crit}} = \frac{1}{4} = 0.25$. We observe that the critical load with this method is quite low.

Charny and Le Boudec prove that obtaining a higher u_{crit} requires taking into account either the peak-rate limitation of the transmission links (*i.e.*, the line-shaping effect) [Le Boudec, Thiran 2001, Thm. 2.4.2] or more details on the paths of the flows or on the topology [Charny, Le Boudec 2000, §4]. In [Le Boudec, Thiran 2001, §6.4.1], a previous result from [Tassiulas, Georgiadis 1994] is extended, and any unidirectional ring of rate-latency servers is proved to be stable. In [Chlamtac, *et al.* 1998], a sufficient stability condition, which is based on the number of interactions between the flows, is provided for homogeneous networks. The condition is extended to general networks in [Rizzo, Le Boudec 2007; Rizzo, Le Boudec 2008]. In [Amari, Mifdaoui 2017], the *Pay Multiplexing Only at Convergence Points* (PMOC) approach is developed and relies on global interference equations.

Many other approaches first rely on an existing method that, tailored for feed-forward networks, is then adapted for the network with cyclic dependencies by using the time-stopping method introduced in [Cruz 1991b]. In a general trend, the more precise the underlying feed-forward method is (by modeling precisely the network, its topology, its elements and its flows), the higher the critical load is for the derived cyclic-dependency method. For example, with the fixed-point problem formulation [Bouillard, Boyer, Le Corronc 2018, Chap. 12], the original network is transformed into a feed-forward network using *cuts*. Then a feed-forward TFA method is iteratively applied to different versions of this feed-forward network. If the iterations meet some convergence conditions, then the overall method can conclude that the original network with cyclic dependencies is stable and latency bounds can be derived. Using this approach, we derived the FP-TFA tool presented in [Thomas, Le Boudec, Mifdaoui 2019]. An overview of FP-TFA is provided in Section 3.3.

We note that a recent work [Plassart, Le Boudec 2021] is based on our results in this chapter. It provides more results on the fixed-point approaches and proves the equivalence of its different versions. Also, in a recent work [Bouillard 2022], Bouillard extends her and Stea's previous work on the linear programming approaches of feed-forward networks [Bouillard, Stea 2015] to include the line-shaping effect and extend the linear programming approach to networks with cyclic dependencies.

Overall, the mathematical approaches for solving the cyclic dependencies take advantage of the line-shaping effect [Grieu 2004; Mifdaoui, Leydier 2017], but they suffer from the burst-propagation phenomenom that creates the cyclic dependencies. They do not require any change in the paths of the flows, but they cannot guarantee the stability of the network *a priori* and usually provide pessimistic delay bounds [Amari, Mifdaoui 2017].

For a given network, choosing between the routing constraints for removing the cyclic dependencies and the mathematical approaches for keeping them is known as the *Break or Solve* dilemma, whose answer depends on many different factors [Finzi, Craciunas 2019].

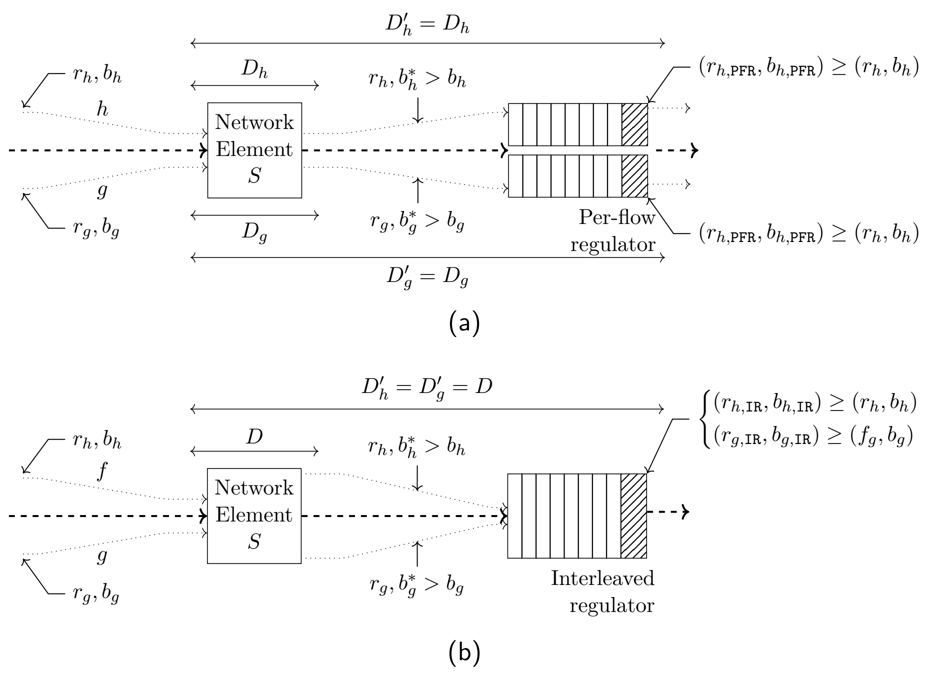


Figure 3.3: Shaping-for-free property of two types of regulators. (a) The per-flow regulator uses one FIFO queue per flow (2 here). The shaping is "for free" for each individual flow. (b) The interleaved regulator uses only one FIFO queue. If the network element S is FIFO for the aggregate $\{h, g\}$, then the shaping is "for free" for the aggregate and both flows have the same delay bound through the concatenation made of S followed by the interleaved regulator.

3.1.5 Using Traffic Regulators to Break the Cyclic Dependencies

A new approach for breaking cyclic dependencies was proposed with the use of traffic regulators [Le Boudec 2018] deployed at every node [Mohammadpour, *et al.* 2018]. Traffic regulators block burstiness propagation by delaying some of the flow's packets.

What are regulators? Regulators come in two flavours: per-flow regulators (PFRs) and interleaved regulators (IRs). A PFR, with parameters $(r_{f,\mathtt{PFR}}, b_{f,\mathtt{PFR}})$ for a flow f, buffers the data of the flow in a FIFO queue (one per flow) and releases the packets, as early as possible while ensuring that the output of this flow f is $\gamma_{r_{f,\mathtt{PFR}},b_{f,\mathtt{PFR}}}$-constrained: Over any window of duration τ, no more than $r_{f,\mathtt{PFR}}\tau + b_{f,\mathtt{PFR}}$ bits of the flow exit the regulator. We say that the regulator *shapes* the flow f with the *shaping curve* $\sigma_{f,\mathtt{PFR}} = \gamma_{r_{f,\mathtt{PFR}},b_{f,\mathtt{PFR}}}$.

A well-known implementation is Linux's Token-Bucket Filter [Wagner 2001]. The PFR can delay some packets, however, it does not increase the *worst-case* delay [Le Boudec 2018], [Le Boudec, Thiran 2001, Thms 1.5.2 and 1.7.3] ("shaping for free" property). This is illustrated in Figure 3.3a. All regulated flows are treated differently from the PFR perspective and can come from a different FIFO system. If the PFR is configured with a rate $r_{f,\mathtt{PFR}}$ and a burst $b_{f,\mathtt{PFR}}$ that are equal or greater than the rate and burst of the flow at the input of the FIFO system, then the shaping-for-free property holds: The overall worst-case delay D'_h for flow h equals its worst-case delay D_h in the FIFO system (and $D'_g = D_g$ as well). The shaping-for-free property of the PFR can be proved using the service-curve characterization of the PFR

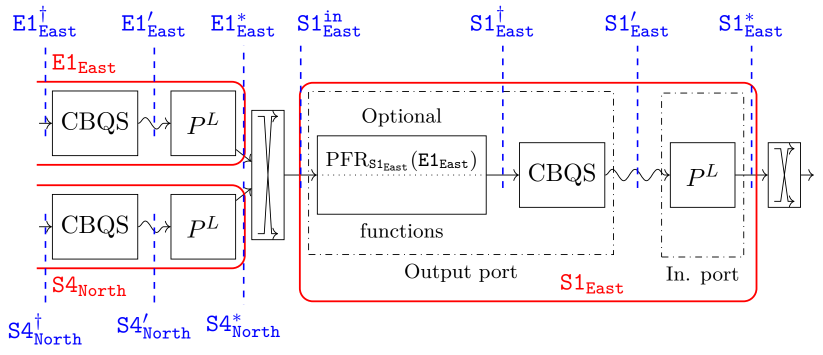

$\mathtt{E1}^{\dagger}_{\mathtt{East}}$
$\mathtt{E1}'_{\mathtt{East}}$
$\mathtt{E1}^{*}_{\mathtt{East}}$
$\mathtt{S1}^{\mathtt{in}}_{\mathtt{East}}$
$\mathtt{S1}^{\dagger}_{\mathtt{East}}$
$\mathtt{S1}'_{\mathtt{East}}$
$\mathtt{S1}^{*}_{\mathtt{East}}$
$\mathtt{E1}_{\mathtt{East}}$
CBQS
P^L
CBQS
P^L
Optional
$\mathrm{PFR}_{\mathtt{S1}_{\mathtt{East}}}(\mathtt{E1}_{\mathtt{East}})$
functions
CBQS
P^L
Output port
$\mathtt{S1}_{\mathtt{East}}$
In. port
$\mathtt{S4}_{\mathtt{North}}$
$\mathtt{S4}^{\dagger}_{\mathtt{North}}$
$\mathtt{S4}'_{\mathtt{North}}$
$\mathtt{S4}^{*}_{\mathtt{North}}$

Figure 3.4: Detailed model of the vertex $\mathtt{S1}_{\mathtt{East}}$ with a PFR installed as an optional function. It processes only the flows coming from parent $\mathtt{E1}_{\mathtt{East}}$.

Proposition 3.1 (Service-curve characterization of the PFR Le Boudec, Thiran 2001, Thm. 1.7.3)
Consider a PFR for a flow f, configured with the shaping curve σ. If f is packetized at the input of the PFR and if σ is a sub-additive curve and $\lim_{t\to 0, t>0}\sigma(t)$ is greater than the maximum packet size of f, then the PFR can be modeled as the concatenation of a greedy-shaper C_σ (Definition 2.7), followed by a packetizer P^L (Definition 2.8).

Interleaved regulators were introduced by [Specht, Samii 2016] (under the name of "Urgency Based Scheduler", also called "Asynchronous Traffic Shaping" by IEEE TSN) in an effort to reduce the required hardware, with respect to the PFR solution. As illustrated in Figure 3.3b, an IR has a single FIFO queue for all the flows it regulates (but every flow f has its own regulation parameter $(r_{f,\mathtt{IR}}, b_{f,\mathtt{IR}})$). The IR examines only the packet at the head of its FIFO queue and releases it, as soon as so doing does not violate the constraint of this flow. Packets of other flows can thus be delayed by the packet at the head of the queue. Nonetheless, an IR that is placed after a FIFO system does not increase the worst-case delay of the FIFO system, as illustrated in Figure 3.3b [Le Boudec 2018, Thm. 4]. Note that this "shaping-for-free" property of the IR holds only if all the flows processed by the IR come from the same previous FIFO system. Also, unlike the PFR, no service-curve characterization is known for the IR.

Where are they located? Regulators are *optional functions* that can be placed just before the class-based queing subsystem (CBQS) of each vertex: Figure 3.4 recalls that the optional functions within vertex $\mathtt{S1}_{\mathtt{East}}$ are located between the observation points $\mathtt{S1}^{\mathtt{in}}_{\mathtt{East}}$ and $\mathtt{S1}^{\dagger}_{\mathtt{Eeast}}$. In this chapter, we assume that each regulator within a vertex regulates only the flows coming from a specific parent of the vertex. For example, Figure 3.4 shows a PFR placed within $\mathtt{S1}_{\mathtt{East}}$ but that processes only the flows coming from $\mathtt{E1}_{\mathtt{East}}$: we use the notation $\mathrm{PFR}_{\mathtt{S1}_{\mathtt{East}}}(\mathtt{E1}_{\mathtt{East}})$ for such PFR. For each processed flow h, its arrival curve at $\mathtt{S1}^{\dagger}_{\mathtt{East}}$ equals the shaping curve configured on the regulator for this flow: $\alpha_{h,\mathtt{S1}^{\dagger}_{\mathtt{East}}} = \sigma_{h,\mathrm{PFR}_{\mathtt{S1}_{\mathtt{East}}}(\mathtt{E1}_{\mathtt{East}})}$. The flows coming from $\mathtt{S4}_{\mathtt{North}}$ are not processed by the PFR. For each non-processed flow g, $\alpha_{g,\mathtt{S1}^{\dagger}_{\mathtt{East}}} = \alpha_{g,\mathtt{S1}^{\mathtt{in}}_{\mathtt{East}}}$

How Can a Regulator Be Deployed? In a *full deployment* approach of the regulators [Mohammadpour, *et al.* 2018; Zhao, Pop, Steinhorst 2021], [Le Boudec, Thiran 2001, end of §6.3.2], the regulators are deployed within all the network's devices. Thus, the burstiness

Table 3.1: Summary of the Regulator Deployment Approaches in Networks with Cyclic Dependencies. Our contributions are highlighted in blue and are compared to the other approaches. As discussed in Section 3.1.3, the solutions that rely on routing constraints are not considered and do not appear in this table.

Regulator deployment approach	Methods and references	
No deployment	Solve cyclic dependencies	
	[Charny, Le Boudec 2000]	
	[Amari, Mifdaoui 2017]	
	[Bouillard, Boyer, Le Corronc 2018, Chap 12]	
	[Finzi, Craciunas 2019]	
	[Bouillard 2022]	
	[Plassart, Le Boudec 2021]	
	Our work, FP-TFA (Section 3.3)	
Full deployment	Per-flow regulators (PFRs)	Interleaved regulators (IRs)
	[Le Boudec, Thiran 2001, §1.7.4]	[Le Boudec 2018]
	[Mohammadpour, *et al.* 2018]	[Mohammadpour, *et al.* 2018]
		[Zhao, Pop, Steinhorst 2021]
Partial deployment	Our work, LCAN (Section 3.2) + FP-TFA (Section 3.3)	

of every flow remains the same along its path; there is no burstiness propagation, and the problems caused by cyclic dependencies are eliminated. Worst-case latencies can be computed using the "shaping-for-free" property and other network-calculus results.

3.1.6 Contributions in This Chapter

The first two rows of Table 3.1 (no-deployment and full-deployment approaches) were the only two options considered in the literature on cyclic dependencies and regulators. In both cases, the stability requirements, cost requirements, and scalability requirements are met only partially.

Evaluation of the Full-Deployment with Respect to the No-Deployment Approach: Our first objective is to evaluate the benefit of the full-deployment approach of either PFRs or IRs on the latency, with respect to the *no-deployment* situation in which the network keeps its cyclic dependencies. To evaluate this benefit, we need a tool that can compute delay bounds, even in networks with cyclic dependencies. The tighter the delay bounds are obtained with this tool, the greater the critical load u_{crit} is.

At the beginning of the PhD, we observed that the recent results of TFA++ [Mifdaoui, Leydier 2017] had not yet been integrated in fixed-point approaches, such as the one developed in [Bouillard, Boyer, Le Corronc 2018, Chap 12]. Therefore, we developed FP-TFA, a tool for computing delay bounds in networks with cyclic dependencies that takes into account the line-shaping effect [Mifdaoui, Leydier 2017].

A Partial Deployment Approach for Regulators by Using LCAN and FP-TFA: As regulators break the propagation of burstiness, we propose an alternative approach by installing only a few regulators within the network, such that all cyclic dependencies are

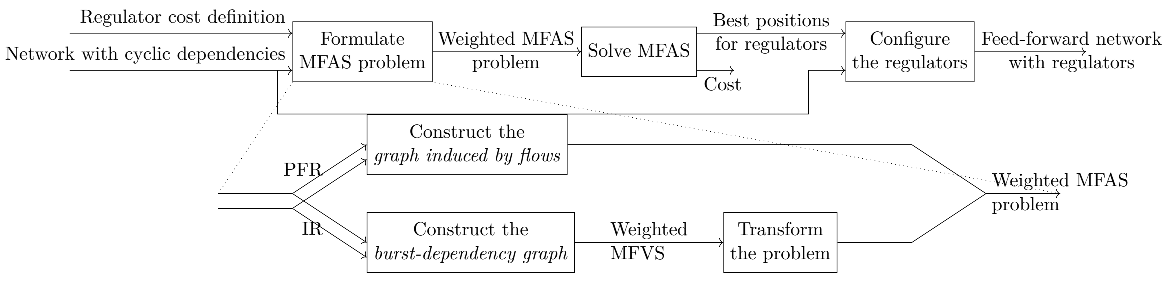

Regulator cost definition
Network with cyclic dependencies
Formulate MFAS problem
Weighted MFAS problem
Solve MFAS
Best positions for regulators
Cost
Configure the regulators
Feed-forward network with regulators
PFR
IR
Construct the *graph induced by flows*
Construct the *burst-dependency graph*
Weighted MFVS
Transform the problem
Weighted MFAS problem

Figure 3.5: Overview of the LCAN algorithm.

removed (*partial deployment*). Hence, the computation of latency bounds is facilitated ($u_{\text{crit}} = 1$) and can be performed by using network calculus and feed-forward methods (Chapter 2). Partial-deployment schemes have not yet been studied and could represent an opportunity for a good compromise among cost, scalability, and performance.

We propose low-cost acyclic network (LCAN), a tool that finds the optimal positions for the regulators and that minimizes the cost of the partial deployment while ensuring that all cyclic dependencies are broken by the regulators. We then use FP-TFA to evaluate the benefit of this partial-deployment approach over the full-deployment and the no-deployment approaches

3.2 LCAN: an Optimal Algorithm for Breaking All Cyclic Dependencies

In this section, we describe low-cost acyclic network (LCAN), an algorithm for breaking all cyclic dependencies through a partial deployment of regulators (either PFRs or IRs). Mixtures of PFRs and IRs, are not considered.

LCAN relies on a user-specified cost function to compute the cost of a regulator. This function can be tailored to reflect the hardware cost of a regulator. Constraints can be added to the algorithm to account for other technical requirements. For example, some switches of the network might not implement at all the regulator option, or some other switches could host only a given maximum number of regulators. These aspects are still under discussion in the real-time community. The external cost function enables LCAN to anticipate the possible outcomes. The recently published standard [IEEE 802.1Qcr] specifies the functional requirements for a bridge that implements the IR model and, with slight modifications, the PFR model. However, it does not discuss the hardware implementation or the associated costs and constraints.

An overview of the LCAN algorithm is presented in Figure 3.5. It first formulates a weighted minimum feedback arc set (MFAS) problem, a well-known NP-complete problem of graph-theory [Karp 1972, 8th item]. The problem is formulated depending on the considered type of regulators for the whole network (either PFR or IR). The output of this step is a directed weighted graph with cycles; the weights represent the configurable costs of the regulators. Second, it uses a state-of-the-art optimal algorithm to solve the MFAS problem [Baharev, Schichl, Neumaier 2015].

In the following section, we describe the problem formulation by using the toy example of

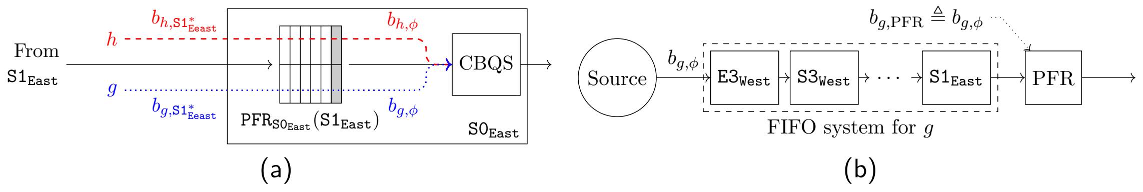

From S1East
h
g
$b_{h,\mathtt{S1}^*_{\mathtt{East}}}$
$b_{g,\mathtt{S1}^*_{\mathtt{East}}}$
$b_{h,\phi}$
$b_{g,\phi}$
$\mathtt{PFR}_{\mathtt{S0}_{\mathtt{East}}}(\mathtt{S1}_{\mathtt{East}})$
CBQS
$\mathtt{S0}_{\mathtt{East}}$
(a)
$b_{g,\mathrm{PFR}} \triangleq b_{g,\phi}$
Source
$b_{g,\phi}$
$\mathtt{E3}_{\mathtt{West}}$
$\mathtt{S3}_{\mathtt{West}}$
$\mathtt{S1}_{\mathtt{East}}$
PFR
FIFO system for g
(b)

Figure 3.6: A PFR is placed for the toy example at $\mathtt{S0}_{\mathtt{East}}$. (a) Content of the output port $\mathtt{S0}_{\mathtt{East}}$: the PFR reshapes all flows coming from $\mathtt{S1}_{\mathtt{East}}$. (b) The system between the source of g and the PFR is FIFO for g, hence the shaping-for-free property holds.

Figure 3.2.

3.2.1 Formulation of the Problem with PFRs

Example: Assume that the output port $\mathtt{S0}_{\mathtt{East}}$ implements a PFR for all the flows coming from $\mathtt{S1}_{\mathtt{East}}$, which we denote by $\mathtt{PFR}_{\mathtt{S0}_{\mathtt{East}}}(\mathtt{S1}_{\mathtt{East}})$. We configure the PFR with the shaping parameters $(r_h, b_{h,\phi})$ and $(r_g, b_{g,\phi})$ *i.e.*, the regulator shapes each flow with the burst that the flow had at its source. Figure 3.6a presents the content of the output port.

The flows competing in the CBQS are h and g with, respectively, arrival bursts $b_{h,\phi}$ and $b_{g,\phi}$. Hence, the computation of the delay bound within the CBQS of $\mathtt{S0}_{\mathtt{East}}$ depends neither on $b_{h,\mathtt{S1}^*_{\mathtt{East}}}$ nor on $b_{g,\mathtt{S1}^*_{\mathtt{East}}}$. The burst propagation from $\mathtt{S1}_{\mathtt{East}}$ to $\mathtt{S0}_{\mathtt{East}}$ is blocked, which corresponds to removing the edge $(\mathtt{S1}_{\mathtt{East}}, \mathtt{S0}_{\mathtt{East}})$ from the GIF in Figure 3.2b.

The shaping-for-free property described in Section 3.1.5 is kept. For the flow g regulated by $\mathtt{PFR}_{\mathtt{S0}_{\mathtt{East}}}(\mathtt{S1}_{\mathtt{East}})$, the FIFO system associated with the regulator is the entire suite of ports from the source of g, up to and including the output port $\mathtt{S1}_{\mathtt{East}}$ (Figure 3.6b).

Finding the optimal positions for the PFRs is hence translated into a MFAS problem: The objective is to remove all the cycles from the GIF (Figure 3.2b) by removing the edges with the smallest weight sum. Each removed edge corresponds to a PFR, and its weight represents the regulator's cost (provided by the external function). In the toy example, if the weights of all edges are equal, then removing edge $(\mathtt{S1}_{\mathtt{East}}, \mathtt{S0}_{\mathtt{East}})$ is an optimal solution to the MFAS problem.

3.2.2 Formulation of the Problem with IRs

There is a significant difference between PFRs and IRs. If an IR is used at a point, say B, in the network to restore the burstiness that exists for a set of flows at some point, say A, the entire path from A to B must be globally FIFO for the aggregate traffic of the set of flows. In practice, this requires that A be a parent (immediate upstream vertex) of B.

Example: Assume that we implement an IR in place of the PFR (Figure 3.7a). We configure[a] the IR such that $b_{h,\mathtt{IR}_{\mathtt{S0}_{\mathtt{East}}}(\mathtt{S1}_{\mathtt{East}})} \triangleq b_{h,\mathtt{S1}^\dagger_{\mathtt{East}}}$ and $b_{g,\mathtt{IR}_{\mathtt{S0}_{\mathtt{East}}}(\mathtt{S1}_{\mathtt{East}})} \triangleq b_{g,\mathtt{S1}^\dagger_{\mathtt{East}}}$: The IR restores the burst at the immediate upstream vertex $\mathtt{S1}_{\mathtt{East}}$ of $\mathtt{S0}_{\mathtt{East}}$ and the system between $\mathtt{S1}^\dagger_{\mathtt{East}}$ and the IR is FIFO for the aggregate $\{f, g\}$ (Figure 3.7b), hence the

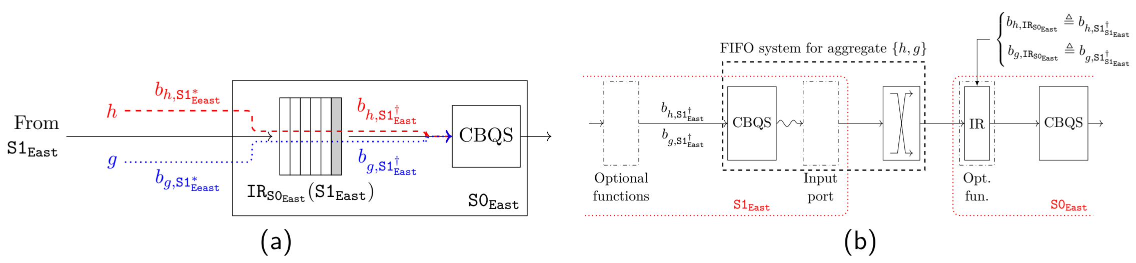

From
$\mathtt{S1_{East}}$
h
g
$b_{h,\mathtt{S1}^*_{\mathtt{East}}}$
$b_{g,\mathtt{S1}^*_{\mathtt{East}}}$
$b_{h,\mathtt{S1}^\dagger_{\mathtt{East}}}$
$b_{g,\mathtt{S1}^\dagger_{\mathtt{East}}}$
CBQS
$\mathtt{IR_{S0_{East}}(S1_{East})}$
$\mathtt{S0_{East}}$
(a)
FIFO system for aggregate $\{h, g\}$
$b_{h,\mathtt{IR_{S0_{East}}}} \triangleq b_{h,\mathtt{S1}^\dagger_{\mathtt{S1_{East}}}}$
$b_{g,\mathtt{IR_{S0_{East}}}} \triangleq b_{g,\mathtt{S1}^\dagger_{\mathtt{S1_{East}}}}$
CBQS
IR
CBQS
Optional functions
Input port
Opt. fun.
$\mathtt{S1_{East}}$
$\mathtt{S0_{East}}$
(b)

Figure 3.7: An IR is placed for the toy example at $\mathtt{S0_{East}}$. (a) Content of the output port $\mathtt{S0_{East}}$: the IR reshapes all flows coming from $\mathtt{S1_{East}}$. (b) The system between $\mathtt{S1}^\dagger_{\mathtt{East}}$ and the input of the IR is FIFO for the aggregate $\{h, g\}$: the shaping-for-free property holds.

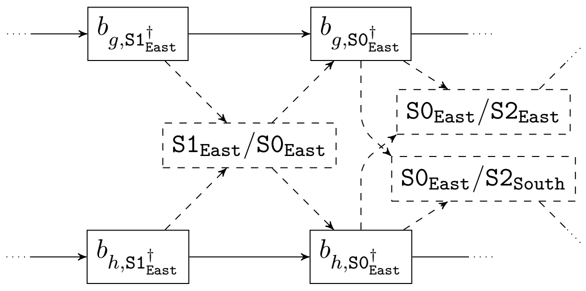

$b_{g,\mathtt{S1}^\dagger_{\mathtt{East}}}$
$b_{g,\mathtt{S0}^\dagger_{\mathtt{East}}}$
$\mathtt{S0_{East}/S2_{East}}$
$\mathtt{S1_{East}/S0_{East}}$
$\mathtt{S0_{East}/S2_{South}}$
$b_{h,\mathtt{S1}^\dagger_{\mathtt{East}}}$
$b_{h,\mathtt{S0}^\dagger_{\mathtt{East}}}$

Figure 3.8: Excerpt of the burst-dependency graph of the toy example.

shaping-for-free property holds.

If the output port of the upstream vertex $\mathtt{S1_{East}}$ does not contain any regulator that processes h, then $b_{h,\mathtt{S1}^\dagger_{\mathtt{East}}} = b_{h,\mathtt{S1}^{\mathtt{in}}_{\mathtt{East}}} = b_{h,\mathtt{E1}^*_{\mathtt{East}}}$, *i.e.*, the burst parameter $b_{h,\mathtt{IR}}$ of the IR within $\mathtt{S0_{East}}$ is equal to the burst of the flow at the output of the vertex $\mathtt{E1_{East}}$. If the output port of the previous vertex $\mathtt{S1_{East}}$ contains an IR that processes flow h, then we simply reuse the configuration of the previous IR:

$$b_{h,\mathtt{IR_{S0_{East}}(S1_{East})}} = b_{h,\mathtt{S1}^\dagger_{\mathtt{East}}} = b_{h,\mathtt{IR_{S1_{East}}(E1_{East})}}$$

In both cases, none of the IR's parameters $b_{h,\mathtt{IR_{S0_{East}}(S1_{East})}}$ or $b_{g,\mathtt{IR_{S0_{East}}(S1_{East})}}$ is entirely independent from the past vertices: they depend on the bounds for the bursts that the flows had in previous hops. As such, the action of the IR, configured as above, cannot be modeled by removing the edge $(\mathtt{S1_{East}}, \mathtt{S0_{East}})$ from the graph in Figure 3.2b.

[a]Notation "≜" means "is equal by definition".

To model the action of the IR, we define a new graph that we call the *burst-dependency graph*. This graph is obtained from the GIF (Figure 3.2b) by using Algorithm 1. For any directed edge $(n \to m)$ in the GIF, Algorithm 1 creates one vertex of the same name ("n/m") for the *burst-dependency graph* (Line 4).

Example: An excerpt of the burst-dependency graph for the toy example is shown in Figure 3.8. The directed edge $(\mathtt{S1_{East}} \to \mathtt{S0_{East}})$ of the GIF in Figure 3.2b is represented, in the *burst-dependency graph*, by the vertex "$\mathtt{S1_{East}/S0_{East}}$" in Figure 3.8.

We call such a vertex a *contention vertex* because it represents the contention that occurs

Algorithm 1 Algorithm that creates the *burst-dependency graph* using the *graph induced by flows.*

```
Require: $\mathcal{G}$ the network's graph induced by flows (GIF). $F$ the set of flows.
Require: For each flow $f$, $\mathcal{G}(f)$ is its flow graph.
1: procedure GetBurstDependencyGraph($\mathcal{G}$)
2:     $\mathcal{V}' \leftarrow \emptyset$, $\mathcal{E}' \leftarrow \emptyset$
3:     for $(n, m) \in$ edges($\mathcal{G}$) do
4:         add a vertex called "$n/m$" to $\mathcal{V}'$                    ▷ Contention vertex: $n/m$
5:     end for
6:     for $f \in F$ do
7:         for $(n, m) \in$ edges($\mathcal{G}(f)$) do
8:             if not already added, add a vertex called "$b_{f,n^\dagger}$" and a vertex "$b_{f,m^\dagger}$" to $\mathcal{V}'$
9:                                              ▷ Burst vertices: $\boxed{b_{f,n^\dagger}}$ and $\boxed{b_{f,m^\dagger}}$
10:            add the edge $(b_{f,n^\dagger}, b_{f,m^\dagger})$ to $\mathcal{E}'$          ▷ Propagation edge: $\longrightarrow$
11:            for all children $m'$ of $n$ in $\mathcal{G}$ do
12:                add the edge $(b_{f,n^\dagger}, n/m')$ to $\mathcal{E}'$           ▷ Contention edges: $\dashrightarrow$
13:            end for
14:            add the edge $(n/m, b_{f,m^\dagger})$ to $\mathcal{E}'$               ▷ Contention edges: $\dashrightarrow$
15:        end for
16:    end for
17:    $\mathcal{G}' \leftarrow (\mathcal{V}', \mathcal{E}')$ return $\mathcal{G}'$
18: end procedure
```

in $\texttt{S1}_{\texttt{East}}$ between the flows of the class of interest and the fact that this contention influences the contention in the next node, $\texttt{S0}_{\texttt{East}}$. If no such contention existed, or if it did not influence the contention in the downstream node $\texttt{S0}_{\texttt{East}}$, such a vertex would not exist in the graph.

Then, for any flow f and any vertex n, Algorithm 1 creates on Line 9 one vertex named "$b_{f,n^\dagger}$" in the *burst-dependency graph.* This vertex represents the value of the burst of f at the observation point $n^\dagger$. Such a vertex is called a *burst vertex.*

Example: In the excerpt of Figure 3.8, we observe the vertex $b_{h,\texttt{S1}^\dagger_{\texttt{East}}}$ that represents the value of the burst of h at the observation point $\texttt{S1}^\dagger_{\texttt{East}}$. We also note the burst vertex $b_{g,\texttt{S1}^\dagger_{\texttt{East}}}$ that represents the value of the burst of the other flow g at the same point.

As the name suggests, the directed edges in the *burst-dependency graph* represent the relation "depends on".

Example: If we focus on the burst vertex $b_{h,\texttt{S0}^\dagger_{\texttt{East}}}$, we note two incoming edges that translate as follows: $b_{h,\texttt{S0}^\dagger_{\texttt{East}}}$, depends on $b_{h,\texttt{S1}^\dagger_{\texttt{East}}}$, the burst that the flow had at $\texttt{S1}^\dagger_{\texttt{East}}$, and on "$\texttt{S1}_{\texttt{East}}/\texttt{S0}_{\texttt{East}}$", *i.e.*, on the contention that occurs within $\texttt{S1}_{\texttt{East}}$.

For the vertex "$\texttt{S1}_{\texttt{East}}/\texttt{S0}_{\texttt{East}}$", the incoming edges can be translated as follows: The level of contention within $\texttt{S1}_{\texttt{East}}$ depends on the the burstiness of the competing flows at $\texttt{S1}^\dagger_{\texttt{East}}$, hence it depends on $b_{h,\texttt{S1}^\dagger_{\texttt{East}}}$ and $b_{g,\texttt{S1}^\dagger_{\texttt{East}}}$ (the two incoming edges). This contention has an effect on the burstiness of the flows that continue towards $\texttt{S0}$: This

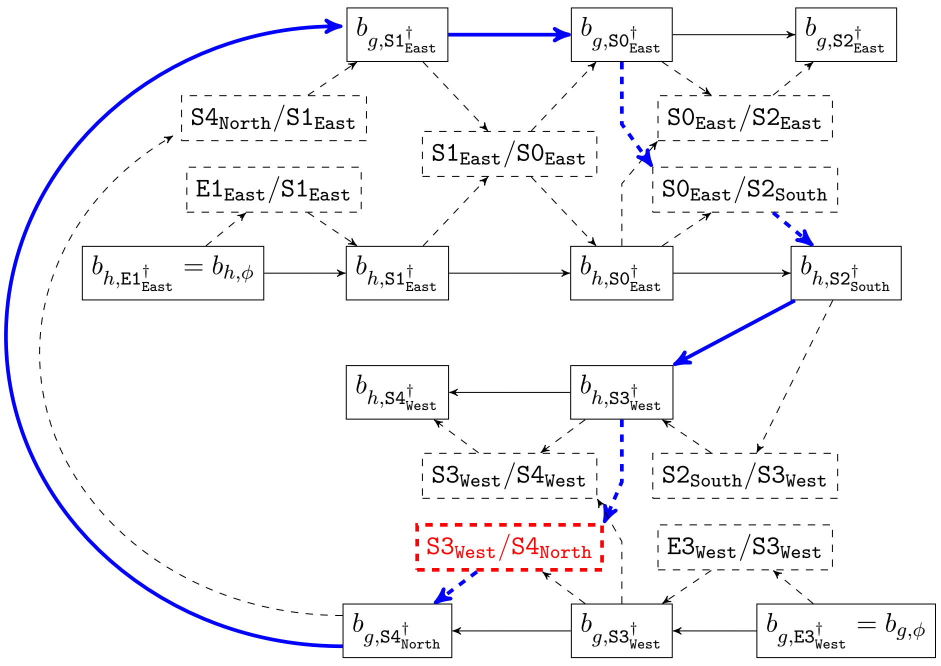

$b_{g,\texttt{S1}^\dagger_{\texttt{East}}}$
$b_{g,\texttt{S0}^\dagger_{\texttt{East}}}$
$b_{g,\texttt{S2}^\dagger_{\texttt{East}}}$
$\texttt{S4}_{\texttt{North}}/\texttt{S1}_{\texttt{East}}$
$\texttt{S1}_{\texttt{East}}/\texttt{S0}_{\texttt{East}}$
$\texttt{S0}_{\texttt{East}}/\texttt{S2}_{\texttt{East}}$
$\texttt{E1}_{\texttt{East}}/\texttt{S1}_{\texttt{East}}$
$\texttt{S0}_{\texttt{East}}/\texttt{S2}_{\texttt{South}}$
$b_{h,\texttt{E1}^\dagger_{\texttt{East}}} = b_{h,\phi}$
$b_{h,\texttt{S1}^\dagger_{\texttt{East}}}$
$b_{h,\texttt{S0}^\dagger_{\texttt{East}}}$
$b_{h,\texttt{S2}^\dagger_{\texttt{South}}}$
$b_{h,\texttt{S4}^\dagger_{\texttt{West}}}$
$b_{h,\texttt{S3}^\dagger_{\texttt{West}}}$
$\texttt{S3}_{\texttt{West}}/\texttt{S4}_{\texttt{West}}$
$\texttt{S2}_{\texttt{South}}/\texttt{S3}_{\texttt{West}}$
$\texttt{S3}_{\texttt{West}}/\texttt{S4}_{\texttt{North}}$
$\texttt{E3}_{\texttt{West}}/\texttt{S3}_{\texttt{West}}$
$b_{g,\texttt{S4}^\dagger_{\texttt{North}}}$
$b_{g,\texttt{S3}^\dagger_{\texttt{West}}}$
$b_{g,\texttt{E3}^\dagger_{\texttt{West}}} = b_{g,\phi}$

Figure 3.9: The *burst-dependency graph* for the toy example. This graph is not acyclic (a cycle is shown in blue), which reflects the fact that the network has cyclic dependencies and is not feed-forward. The possible positions for interleaved regulators are shown with dashed vertices. Placing in the network an IR at $\texttt{S4}_{\texttt{North}}$ in order to regulate all flows coming from its parent $\texttt{S3}_{\texttt{West}}$ corresponds to removing the red vertex $\texttt{S3}_{\texttt{West}}/\texttt{S4}_{\texttt{North}}$ from the graph, which makes it acyclic. Therefore, the $\texttt{IR}_{\texttt{S4}_{\texttt{North}}}(\texttt{S3}_{\texttt{West}})$ removes all the cyclic dependencies from the network. Here, $\texttt{IR}_{\texttt{S4}_{\texttt{North}}}(\texttt{S3}_{\texttt{West}})$ is the optimal position for an IR for removing all the cyclic dependencies from the network with as few IRs as possible.

affects $b_{h,\texttt{S0}^\dagger_{\texttt{East}}}$ and $b_{g,\texttt{S0}^\dagger_{\texttt{East}}}$ (the two outgoing edges).

The *burst-dependency graph* can be seen as a graph that is more precise than the GIF: In the *burst-dependency graph*, we explicit the contention in every vertex, as well as the burst of the flows that participate in each contention.

Example: The full version of the *burst-dependency* graph for the toy example is given in Figure 3.9. This graph is not acyclic (a cycle is highlighted in thick blue), which is consistent with the fact that the network has cyclic dependencies (as identified previously).

If we place a regulator $\texttt{IR}_n(m)$ at vertex n (*i.e.*, just before its CBQS) for processing the flows coming from its parent m, for each processed flow f, the burst $b_{f,n^\dagger}$ of f at $n^\dagger$ equals $b_{f,m^\dagger}$ and does not depend on the contention that occured within m. We model this by keeping the edge $b_{f,m^\dagger} \to b_{f,n^\dagger}$ in the *burst-dependency graph*, but removing the edge "m/n" $\to b_{f,n^\dagger}$. Doing this for all flows $f \ni (m,n)$ is equivalent to removing entirely the vertex "m/n" with all its incoming and outgoing edges.

Example: If we place an interleaved regulator $\texttt{IR}_{\texttt{S0}_{\texttt{East}}}(\texttt{S1}_{\texttt{East}})$ within $\texttt{S0}_{\texttt{East}}$ that processes all flows coming from $\texttt{S1}_{\texttt{East}}$, then the configuration that ensures the shaping-for-free property is $b_{h,\texttt{IR}_{\texttt{S0}_{\texttt{East}}}(\texttt{S1}_{\texttt{East}})} \triangleq b_{h,\texttt{S1}^\dagger_{\texttt{East}}}$ and $b_{g,\texttt{IR}_{\texttt{S0}_{\texttt{East}}}(\texttt{S1}_{\texttt{East}})} \triangleq b_{g,\texttt{S1}^\dagger_{\texttt{East}}}$. Thus, the burst $b_{h,\texttt{S0}^\dagger_{\texttt{East}}}$ of the flow h at $\texttt{S0}^\dagger_{\texttt{East}}$ depends only on its previous burst $b_{h,\texttt{S1}^\dagger_{\texttt{East}}}$, and similarly for g. This corresponds to removing the *contention vertex* $\texttt{S1}_{\texttt{East}}/\texttt{S0}_{\texttt{East}}$ (and all its edges) from the burst-dependency graph in Figures 3.8 and 3.9.

Finding the optimal positions for the IRs is hence translated into a minimum feedback vertex set (MFVS) problem formulation: The objective is to remove the cycles from the *burst-dependency graph* by removing as few *contention vertices* as possible. Each removed *contention vertex* represents an IR and the weight of each vertex is the configurable cost of the IR, provided by the external cost function.

Example: With Figure 3.9, we observe that the $\texttt{IR}_{\texttt{S0}_{\texttt{East}}}(\texttt{S1}_{\texttt{East}})$ is not sufficient to solve the MFVS problem and to obtain a feed-forward network. Indeed, removing the contention vertex $\texttt{S1}_{\texttt{East}}/\texttt{S1}_{\texttt{East}}$ does not break the blue cycle.

If all the possible positions have the same cost, then placing an IR at $\texttt{IR}_{\texttt{S4}_{\texttt{North}}}(\texttt{S3}_{\texttt{West}})$ is the optimal solution for breaking all cyclic dependencies in the toy example: This corresponds to removing the *contention vertex* $\texttt{S3}_{\texttt{West}}/\texttt{S4}_{\texttt{North}}$ (in thick red) from the graph in Figure 3.9, which makes it acyclic.

In graph theory, MFVS and MFAS are equivalent problems, and the former can be transformed into the latter with minimal graph manipulations [Baharev, Schichl, Neumaier 2015]. This transformation is performed by the *transform the problem* step of Figure 3.5.

3.2.3 Solving the MFAS Problem

Finding a minimum feedback arc set (MFAS) is a well-known NP-complete problem [Garey, Johnson 1990; Karp 1972]. We solve it by using the algorithm proposed in [Baharev, Schichl, Neumaier 2015]. This algoritm provides an optimal solution and is well suited for graphs containing up to one million cycles, a reasonable limit for industrial cases.

3.3 FP-TFA: Computing Tight Delay Bounds in Networks with Cyclic Dependencies

LCAN can be used to achieve a partial deployment of either IRs or PFRs. To compare the deterministic performances of the partial-deployment approach against the no-deployment and the full-deployment approaches (first two rows of Table 3.1), we need an algorithm for computing delay bounds in networks that can contain cyclic dependencies and/or regulators.

In [Thomas, Le Boudec, Mifdaoui 2019], we present FP-TFA, a tool for computing delay bounds in networks with cyclic dependencies. Since the publication of [Thomas, Le Boudec, Mifdaoui 2019], we have developed a new version of the tool, called *experimental modular TFA* (xTFA), that provides more flexibility for the models and includes our other theoretical contributions (Chapters 4 and 5). The algorithmic aspects of xTFA are presented in detail in Chapter 6. In the current section, we focus only on the main theoretical results that support FP-TFA.

FP-TFA is a tool based primarly on the TFA++ algorithm proposed by Mifdaoui and Leydier [Mifdaoui, Leydier 2017] that is itself based on the TFA compositional approach presented in Section 2.5.1. FP-TFA adds to TFA++ (a) a new result on the effect of the packetizers, (b) the recent improvement proposed in [Mohammadpour, Stai, Le Boudec 2019], (c) the model of the PFRs and IRs (as configured by LCAN), and (d) an extension of TFA++

to topologies with cyclic dependencies by relying on the fixed-point approach of [Bouillard, Boyer, Le Corronc 2018, Chap. 12], but with a new result on the validity of the fixed-point approach.

3.3.1 A Novel Result on the Effect of Packetizers

Each input port in the device model used throughout the thesis (Section 2.6) contains a network-calculus packetizer (Definition 2.8) that does not increase the per-packet delay bounds of the upstream system (Theorem 2.5). However, it does increase the burstiness of the flows at its output. This increase is known to be bounded by $l_{\max}$, the maximum packet length of the flow (Section 2.3.3).

In [Thomas, Le Boudec, Mifdaoui 2019], we use the duality between max-plus and min-plus constraints to provide a better bound for the burstiness increase, when the input of the packetizer is connected to a transmission link with fixed capacity, as it is the case in our device model (Figure 2.10). Here, we provide a stronger result in the form of a service curve, from which the packetizer burstiness-increase bound of [Thomas, Le Boudec, Mifdaoui 2019] can be retrieved.

Theorem 3.1 (Service curve of a packetizer with a known input rate)
A packetizer P^L placed on a line of fixed transmission rate c provides for every flow the service curve $\delta_{\frac{l_{\max}}{c}}$, where $l_{\max}$ is the maximum packet length of the flow.

The proof is in Appendix B.1.1. If $\gamma_{r,b}$ is a leaky-bucket arrival curve for a flow f at the input of a packetizer, then applying Theorem 2.2 with the above service curve gives that $\gamma_{r,b+l_{\max}\frac{r}{c}}$ is an arrival curve for the flow at the packetizer output, which we use in FP-TFA for computing the effect of the packetizer on the aggregate arrival curves. The tool continues to use Theorem 2.5 for the effect of the packetizer on the latency bounds.

3.3.2 Delay Bound within a Vertex

Arrival Curve for the Aggregate at the Input of the CBQS

We now rely on the TFA model of the FIFO policy (Section 2.4.2) and, to compute delay bounds within a given vertex, we combine the new result on the packetizer with the improvement proposed by [Mohammadpour, Stai, Le Boudec 2019].

Consider a vertex n of the network's GIF $\mathcal{G}$ and assume that for each parent p of n in $\mathcal{G}$ and for each flow f such that $f \ni (p,n)$, we know a leaky-bucket arrival curve $\alpha_{f,p'} = \gamma_{r_f,b_{f,p'}}$ for f at the observation point p'.

Example: We consider $n = \texttt{S1}_{\texttt{East}}$ as in Figure 3.10, and we assume that we know a leaky-bucket arrival curve $\alpha_{h,\texttt{E1}'_{\texttt{East}}} = \gamma_{r_h,b_{h,\texttt{E1}'_{\texttt{East}}}}$ [resp., $\alpha_{g,\texttt{S4}'_{\texttt{North}}} = \gamma_{r_g,b_{g,\texttt{S4}'_{\texttt{North}}}}$] for h [resp., g] at observation point $\texttt{E1}'_{\texttt{East}}$ [resp., $\texttt{S4}'_{\texttt{North}}$].

The CBQS within vertex n provides the service curve β_n to the aggregate $\{f | f \ni n\}$ (Section 2.6.4). To apply Proposition 2.3, we compute an arrival curve for the aggregate at the input of the CBQS, *i.e.*, at observation point $n^\dagger$. We can decompose the set of flows

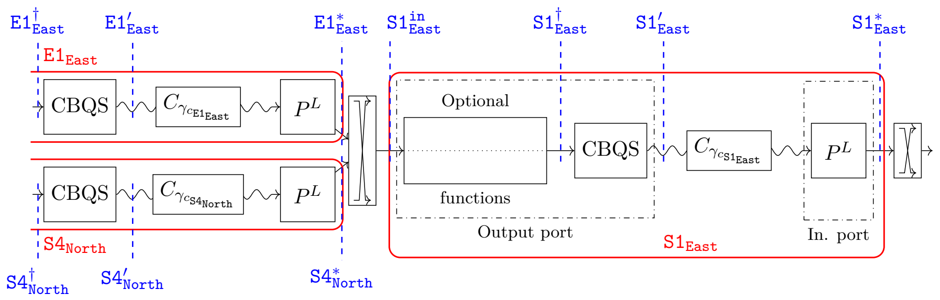


Figure 3.10: Detailed model for the computation of the delay bound through vertex $\texttt{S1}_{\texttt{East}}$ of the toy example. It results from a direct application of the model in Section 2.6 to the toy example of this chapter.

belonging to such aggregate as follows:

$$\{f|f \ni n\} = \{f|\text{source}(f) = n\ \} \cup \left(\bigcup_{p,\ \text{parent of } n \text{ in } \mathcal{G}} \{f|f \ni (p,n)\} \right) \tag{3.1}$$

Example: There exists no flow whose source is $\texttt{S1}_{\texttt{East}}$, thus

$$\begin{aligned} \{f|f \ni \texttt{S1}_{\texttt{East}}\} &= \{f|f \ni (\texttt{E1}_{\texttt{East}}, \texttt{S1}_{\texttt{East}})\} \cup \{f|f \ni (\texttt{S4}_{\texttt{North}}, \texttt{S1}_{\texttt{East}})\} \\ &= \{h\} \cup \{g\} \end{aligned}$$

An arrival curve for the aggregate at $n^\dagger$ is hence

$$\alpha_{n^\dagger} = \sum_{f|\text{source}(f)=n} \alpha_{f,\phi} + \left(\sum_{p,\ \text{parent of } n \text{ in } \mathcal{G}} \alpha_{(p,n),n^\dagger} \right) \tag{3.2}$$

where $\alpha_{(p,n),n^\dagger}$ represents an arrival curve for the sub-aggregate $\{f|f \ni (p,n)\}$ at observation point $n^\dagger$. The left-hand side of (3.2) is directly obtained from the source specifications. If we focus on only the right-hand side, our goal is to obtain, for each parent p, an arrival curve $\alpha_{(p,n),n^\dagger}$. To do so, for a parent p, we start at observation point p', at which we know an arrival curve $\alpha_{(p,n),p'}$ for the sub-aggregate $\{f|f \ni (p,n)\}$, and we model the evolution of the arrival curve for the sub-aggregate, from p' to $n^\dagger$.

Example: We start with $p = \texttt{S4}_{\texttt{North}}$. We seek to obtain an arrival curve $\alpha_{(\texttt{S4}_{\texttt{North}},\texttt{S1}_{\texttt{East}}),\texttt{S1}^\dagger_{\texttt{East}}}$ for the aggregate $\{f|f \ni (\texttt{S4}_{\texttt{North}}, \texttt{S1}_{\texttt{East}})\} = \{h\}$ at observation point $\texttt{S1}^\dagger_{\texttt{East}}$, starting with an arrival curve $\alpha_{(\texttt{S4}_{\texttt{North}},\texttt{S1}_{\texttt{East}}),\texttt{S4}'_{\texttt{North}}}$ of the same aggregate at observation point $\texttt{S4}'_{\texttt{North}}$ (Figure 3.10).

At p', we know by assumption an arrival curve for each flow of the class of interest, hence we have a first arrival curve for the sub-aggregate:

$$\alpha_{(p,n),p'} = \sum_{f|f\ni(p,n)} \alpha_{f,p'} \tag{3.3}$$

When transmitted on the transmission link within p, this sub-aggregate is submitted to the effect of the greedy shaper $C_{\gamma_{c_p,0}}$ representing the transmission link of capacity c_p. The network is underloaded, thus for any flow $f \ni (p,n)$, $\gamma_{c_p,0} \geq \alpha_{f,p'}$. Furthermore, $\gamma_{c_p,0}$ is a sub-additive curve. By application of Theorem 2.4, the greedy shaper does not increase the end-to-end latency bounds and keeps the input arrival-curve constraints. As a result, the sub-aggregate exits the transmission link with the arrival curve

$$\alpha_{(p,n),p'} = \gamma_{c_p} \otimes \left(\sum_{f|f\ni(p,n)} \alpha_{f,p'} \right) \tag{3.4}$$

where $\gamma_{c_p} = \gamma_{c_p,0}$ is a leaky-bucket curve with a rate c_p, the capacity of the link, and a burst 0.

Example: In the toy example, we obtain $\alpha_{(\mathtt{S4}_{\mathtt{North}},\mathtt{S1}_{\mathtt{East}}),\mathtt{S4}'_{\mathtt{North}}} = \gamma_{c_{\mathtt{S4}_{\mathtt{North}}}} \otimes \alpha_{h,\mathtt{S4}_{\mathtt{North}}'}$.

The next step is to obtain an arrival curve at p^*. We use our novel result on the service curve of the packetizer (Theorem 3.1) in order to obtain an arrival curve at the output of the packetizer, P_L^*.

$$\alpha_{(p,n),P_L^*} = \gamma_{c_p,l_{\max}} \otimes \left(l_{\max} \frac{\sum_{f|f\ni(p,n)} r_f}{c_p} + \sum_{f|f\ni(p,n)} \alpha_{f,p'} \right) \tag{3.5}$$

Example: We obtain

$$\alpha_{(\mathtt{S4}_{\mathtt{North}},\mathtt{S1}_{\mathtt{East}}),P_L^*} = \gamma_{c_{\mathtt{S4}_{\mathtt{North}}},l_{\max}} \otimes \gamma_{r_g,b_{g,\mathtt{S4}'_{\mathtt{North}}}+\frac{r_g}{c_{\mathtt{S4}_{\mathtt{North}}}}l_{\max}} \tag{3.6}$$

The resulting arrival curve is a piece-wise linear curve shown in Figure 3.11a.

If the input port has additional technological latencies or if the switching fabric has a latency (as in Figure 2.10), then their corresponding latency bounds worsen the above arrival curve. For illustration purposes, we assume here that no such latencies exist, thus $\alpha_{(p,n),p^*} = \alpha_{(p,n),P_L^*}$, *i.e.*, the output of vertex p is the output of the packetizer P_L. Additionally, consider for now that no regulator has been installed at n. Then $\alpha_{(p,n),n^\dagger} = \alpha_{(p,n),n^{\text{in}}} = \alpha_{(p,n),p^*} = \alpha_{(p,n),P_L^*}$.

Example: If no regulator is installed, either for the sub-aggregate coming from $\mathtt{S4}_{\mathtt{North}}$, or for the sub-aggregate coming from $\mathtt{E1}_{\mathtt{East}}$, then

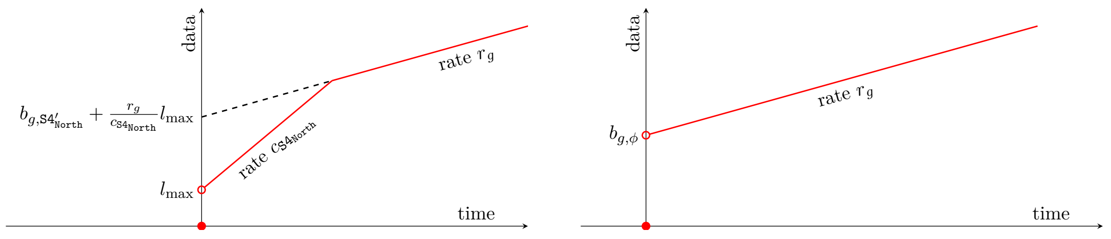
$$\alpha_{(\mathtt{S4}_{\mathtt{North}},\mathtt{S1}_{\mathtt{East}}),\mathtt{S1}^\dagger_{\mathtt{East}}} = \alpha_{(\mathtt{S4}_{\mathtt{North}},\mathtt{S1}_{\mathtt{East}}),\mathtt{S4}^*_{\mathtt{North}}} = \gamma_{c_{\mathtt{S4}_{\mathtt{North}}},l_{\max}} \otimes \gamma_{r_g,b_{g,\mathtt{S4}'_{\mathtt{North}}}+\frac{r_g}{c_{\mathtt{S4}_{\mathtt{North}}}}l_{\max}}$$

And this arrival curve is the same as in Figure 3.11a.

In conclusion, for each parent p of n in $\mathcal{G}$, we obtain $\alpha_{(p,n),n^\dagger}$, an arrival curve for the sub-aggregate $\{f|f \ni (p,n)\}$ at the observation point $n^\dagger$. Applying (3.2), we obtain $\alpha_{n^\dagger}$, an arrival curve for the entire aggregate $\{f|f \ni n\}$.

Example:

$$\alpha_{\mathtt{S1}^\dagger_{\mathtt{East}}} = \left(\gamma_{c_{\mathtt{S4}_{\mathtt{North}}},l_{\max}} \otimes \gamma_{r_g,b_{g,\mathtt{S4}'_{\mathtt{North}}}+\frac{r_g}{c_{\mathtt{S4}_{\mathtt{North}}}}l_{\max}}\right) + \left(\gamma_{c_{\mathtt{E1}_{\mathtt{East}}},l_{\max}} \otimes \gamma_{r_f,b_{f,\mathtt{E1}'_{\mathtt{East}}}+\frac{r_f}{c_{\mathtt{E1}_{\mathtt{East}}}}l_{\max}}\right)$$



(a) (b)

Figure 3.11: Contribution of the parent $\mathtt{S4}_{\mathtt{North}}$ in the aggregate arrival curve at the input of the CBQS within $\mathtt{S1}_{\mathtt{East}}$, *i.e.*, $\alpha_{(\mathtt{S4}_{\mathtt{North}},\mathtt{S1}_{\mathtt{East}}),\mathtt{S1}^{\dagger}_{\mathtt{East}}}$. (a) When no regulator is used, the curve is obtained from (3.5). (b) When a regulator is placed between $\mathtt{S4}_{\mathtt{North}}$ and $\mathtt{S1}_{\mathtt{East}}$, the curve is obtained by summing all the shaping curves of all the regulated flows.

Delay Bound within the CBQS

Finally, we compute the delay bound D_n by using the arrival curve $\alpha_{n^{\dagger}}$ for the aggregate of the class-of-interest at $n^{\dagger}$, β_n, the service curve of the CBQS within n for the class-of-interest, and Proposition 2.3 that also states that $D_{f,n} = D_n$ is also a delay bound through the CBQS for each flow f that belongs to the class of interest.

Then, we apply the delay-bound improvement proposed by [Mohammadpour, Stai, Le Boudec 2019]. We note that Daigmorte obtains a different yet similar result; he focuses on strict service curves [Daigmorte 2019, Chap. 5]. If β_n is a rate-latency service curve $\beta_n = \beta_{R_n,T_n}$ with a rate R_n and a latency T_n, then

$$D^*_{n,f} = D_{n,f} - l_{\min,f}\left(\frac{1}{R_n} - \frac{1}{c_n}\right)$$

is also a delay bound for f through the CBQS, where $l_{\min,f}$ is the minimal packet size of the flow [Mohammadpour, Stai, Le Boudec 2019]. Last, by property of the packetizer (Theorem 2.5), $D^*_{n,f}$ is also a delay bound for f through the combined system made of the CBQS and the packetizer.

Computing the Individual Arrival-Curves

For each flow f of the class-of-interest crossing n, we compute the arrival curves for this individual flow at the different observation points $w \in \{n^{\mathtt{in}}, n^{\dagger}, n', n^*\}$ by considering each time the system $S_{p',w}$ made of all network elements between the observation point p' and the observation point w, where p is the parent of n from which f arrives ($f \ni (p,n)$). We obtain $D_f^{p' \to w}$, a delay upper bound for f between p' and w by summing all maximal technological latencies applicable located between p' and w, plus $D^*_{f,n}$ if $w \in \{n', n^*\}$. We do the same with all minimum technological latencies to obtain $d_f^{p' \to w}$, a lower bound for the delay of f through $S_{p',w}$.

Depending on w, $S_{p',w}$ can contain network elements such as an input port, a transmission link, an output port and an (input, output) pair of a switching fabric. As discussed in Section 2.6, each of them is causal, lossless and FIFO, hence $S_{p',w}$ is also a causal, lossless, FIFO system for f with a delay for f bounded within $[d_f^{p'\to w}, D_f^{p'\to w}]$. Thus, $\alpha_{f,p'} \oslash \delta_{D_f^{p'\to w} - d_f^{p'\to w}}$ is an arrival curve for f at w. This is a leaky-bucket arrival curve from which we obtain the burst bound b_w. Repeating this for all observation points $w \in \{n^{\mathtt{in}}, n^{\dagger}, n', n^*\}$, we obtain for each f, the burst bounds $b_{f,n^{\mathtt{in}}}$, $b_{f,n^{\dagger}}$, $b_{f,n'}, b_{f,n^*}$.

By doing this for all flows and by focusing on the observation point n', the computations performed at vertex n define an application $\mathcal{F}_n$:

$$\mathcal{F}_n : (b_{f,q'})_{\forall q \text{ parent of } n; \forall f | f \ni (q,n)}, (b_{f,\phi})_{\forall f | \text{source}(f)=n} \mapsto (b_{f,n'})_{\forall f | f \ni n} \tag{3.7}$$

which, by using a vector notation, can be written

$$\mathcal{F}_n : \boldsymbol{b_n} \mapsto \boldsymbol{b'_n} \tag{3.8}$$

with $\boldsymbol{b_n} = (b_{f,q'})_{\forall q \text{ parent of } n; \forall f | f \ni (q,n)}, (b_{f,\phi})_{\forall f | \text{source}(f)=n}$ and $\boldsymbol{b'_n} = (b_{f,n'})_{\forall f | f \ni n}$. $\boldsymbol{b_n}$ and $\boldsymbol{b'_n}$ have the same dimension by (3.1).

Computing End-to-End Delay Bounds in Feed-Forward Networks

When the network is feed-forward, the GIF $\mathcal{G}$ is acyclic and we can define a topological order $(n_1, n_2, \ldots, n_z)$ of the vertices. Then, by definition of the order, n_1 has no parents in $\mathcal{G}$, thus the values of the vector $\boldsymbol{b_{n_1}} = (b_{f,\phi})_{\forall f | \text{source}(f)=n_1}$ are known by assumption. We can chain the $\mathcal{F}_n$ applications until all vertices have been computed.

During this process, we store the delay bounds $d_f^{p'\to n'}$ and $D_f^{p'\to n'}$ for each vertex n, each of parent p and each flow f. When all the vertices have been computed, the end-to-end delay upper- and lower-bounds for a flow f are obtained by summing the per-vertex delay bounds along the path of the flow.

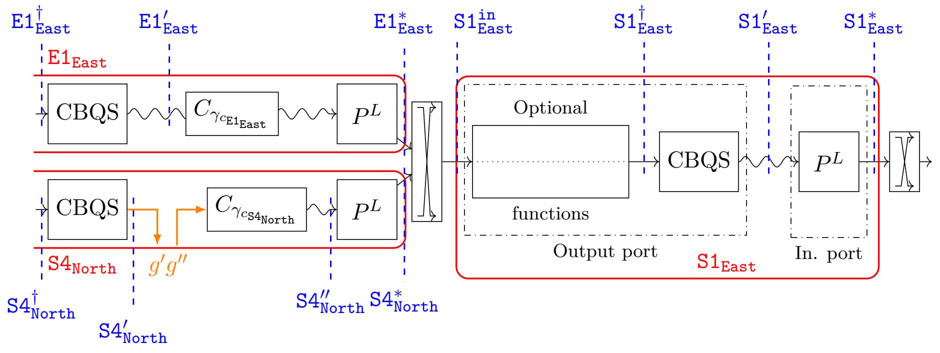
3.3.3 Extension of the FP-TFA Approach to Networks with Cyclic Dependencies

When the network contains cyclic dependencies, it is impossible to find a starting vertex among the vertices that are part of a cycle.

Example: In the toy example, none of $\mathtt{S1}_{\mathtt{East}}, \mathtt{S0}_{\mathtt{East}}, \mathtt{S2}_{\mathtt{South}}, \mathtt{S3}_{\mathtt{West}}$ and $\mathtt{S4}_{\mathtt{North}}$ can be selected as a starting point because they belong to the same cycle.

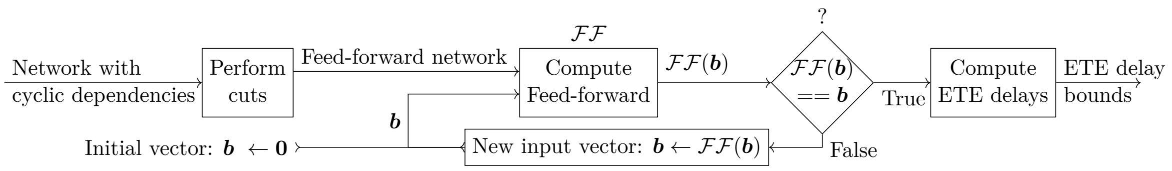
In order to compute end-to-end delay bounds in networks with cyclic dependencies, the idea is to virtually perform cuts in the topology so as to make it acyclic [Bouillard, Boyer, Le Corronc 2018]. A cut here is a virtual disconnection of a transmission link that disconnects a parent from one of its children. The cuts can be selected freely, as long as the remaining network is feed-forward. In the FP-TFA tool, we use LCAN to identify them: The fewer cuts are selected, the less memory is used to keep track of their related variables.

Example: We select the cut $(\mathtt{S4}_{\mathtt{North}}, \mathtt{S1}_{\mathtt{East}})$. This cut is sufficient to remove all cyclic dependencies. In the resulting virtual network, everything occurs for vertex $\mathtt{S1}_{\mathtt{East}}$ as if the transmission link in its parent $\mathtt{S4}_{\mathtt{North}}$ has been disconnected. In particular, it splits the flow g into two virtual subflows: g' upstream of the cut and g'' after the cut (Figure 3.12).



$\mathtt{E1}^{\dagger}_{\mathtt{East}}$
$\mathtt{E1}'_{\mathtt{East}}$
$\mathtt{E1}^{*}_{\mathtt{East}}$
$\mathtt{S1}^{\mathtt{in}}_{\mathtt{East}}$
$\mathtt{S1}^{\dagger}_{\mathtt{East}}$
$\mathtt{S1}'_{\mathtt{East}}$
$\mathtt{S1}^{*}_{\mathtt{East}}$
$\mathtt{E1}_{\mathtt{East}}$
CBQS
$C_{\gamma_{c_{\mathtt{E1East}}}}$
P^L
Optional
functions
CBQS
P^L
CBQS
$C_{\gamma_{c_{\mathtt{S4North}}}}$
P^L
$\mathtt{S4}_{\mathtt{North}}$
$g'\,g''$
Output port
$\mathtt{S1}_{\mathtt{East}}$
In. port
$\mathtt{S4}^{\dagger}_{\mathtt{North}}$
$\mathtt{S4}'_{\mathtt{North}}$
$\mathtt{S4}''_{\mathtt{North}}$
$\mathtt{S4}^{*}_{\mathtt{North}}$

Figure 3.12: Virtual cut $(\mathtt{S4}_{\mathtt{North}}, \mathtt{S1}_{\mathtt{East}})$ that disconnects the transmission link within $\mathtt{S4}_{\mathtt{North}}$ on the toy example.



Network with cyclic dependencies
Perform cuts
Feed-forward network
$\mathcal{FF}$
Compute Feed-forward
$\mathcal{FF}(\boldsymbol{b})$
?
$\mathcal{FF}(\boldsymbol{b}) == \boldsymbol{b}$
True
Compute ETE delays
ETE delay bounds
$\boldsymbol{b}$
Initial vector: $\boldsymbol{b} \leftarrow \boldsymbol{0}$
New input vector: $\boldsymbol{b} \leftarrow \mathcal{FF}(\boldsymbol{b})$
False

Figure 3.13: Overview of the FP-TFA algorithm when it searches for a fixed point of the feed-forward algorithm on the virtual feed-forward network.

Given the knowledge of the bursts after the cuts (vector $\boldsymbol{b}$), the iterative application of the $\mathcal{F}_n$ functions provides an algorithm, $\mathcal{FF}$, that computes the delay bounds for the bursts $\boldsymbol{b'}$ just before the cuts: $\mathcal{FF} : \boldsymbol{b} \mapsto \boldsymbol{b'}$.

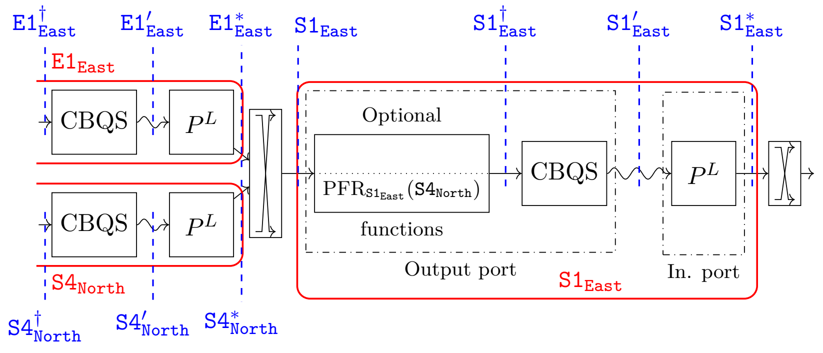
Example: The knowledge of a bound $b_{g'',\mathtt{S4}''_{\mathtt{North}}}$ for the burst of the virtual flow g'' just after the cut enables the computation of the application

$$\mathcal{F}_{\mathtt{S1}_{\mathtt{East}}} : (b_{g'',\mathtt{S4}''_{\mathtt{North}}}, b_{h,\mathtt{E1}'_{\mathtt{East}}}) \mapsto (b_{g'',\mathtt{S1}'_{\mathtt{East}}}, b_{h,\mathtt{S1}'_{\mathtt{East}}})$$

This enables then the computation of $\mathcal{F}_{\mathtt{S0}_{\mathtt{East}}}(b_{g'',\mathtt{S1}'_{\mathtt{East}}}, b_{h,\mathtt{S1}'_{\mathtt{East}}})$ and so on. The iterative computation of $\mathtt{S1}_{\mathtt{East}}, \mathtt{S0}_{\mathtt{East}}, \dots, \mathtt{S4}_{\mathtt{North}}$ hence provides an algorithm $\mathcal{FF}$ that maps the vector $\boldsymbol{b} = (b_{g'',\mathtt{S4}''_{\mathtt{North}}})$ of the burst bound after the cut to the vector $\boldsymbol{b'} = (b_{g',\mathtt{S4}'_{\mathtt{North}}})$ of the burst before the cut.

If the real network, with no cut, is stable, then for every cut tuple (p, n) and every split flow f there exists a lowest possible bound on the burstiness of f at p'. Thus, the vector $\tilde{\boldsymbol{b}}$ of these lowest burst bounds must verify $\mathcal{FF}(\tilde{\boldsymbol{b}}) \geq \tilde{\boldsymbol{b}}$. By the monotonicity of $\mathcal{FF}$, it follows that $\boldsymbol{b}$ is upper-bounded by the (possibly infinite) largest fixed point of $\mathcal{FF}$. In Theorem 2, we prove a stronger result: For any non-negative and finite fixed-point $\overline{\boldsymbol{b}}$ of $\mathcal{FF}$ (i.e., $\mathcal{FF}(\overline{\boldsymbol{b}}) = \overline{\boldsymbol{b}}$) and if the real network is initially empty, then the real network is stable and $\boldsymbol{b} \leq \overline{\boldsymbol{b}}$, i.e., the fixed-point $\overline{\boldsymbol{b}}$ is a valid bound for the burst of the flows at the cuts.

To find such a fixed point, FP-TFA iterates the feed-forward algorithm $\mathcal{FF}$, starting with the empty vector $\boldsymbol{0}$ (Figure 3.13). After each call to $\mathcal{FF}$, we check if the fixed point is reached for the current iteration ($\mathcal{FF}(\boldsymbol{b}) == \boldsymbol{b}$). For the fixed point to be reached, strict equality must be achieved for each term of the vectors, as burst values are in integers (in bits). If the fixed point is not used, the output bursts $\mathcal{FF}(\boldsymbol{b})$ are used as input $\boldsymbol{b}$ to a new call to $\mathcal{FF}$.



$\mathtt{E1}^{\dagger}_{\mathtt{East}}$
$\mathtt{E1}'_{\mathtt{East}}$
$\mathtt{E1}^{*}_{\mathtt{East}}$
$\mathtt{S1}_{\mathtt{East}}$
$\mathtt{S1}^{\dagger}_{\mathtt{East}}$
$\mathtt{S1}'_{\mathtt{East}}$
$\mathtt{S1}^{*}_{\mathtt{East}}$
$\mathtt{E1}_{\mathtt{East}}$
CBQS
P^L
CBQS
P^L
Optional
$\mathrm{PFR}_{\mathtt{S1}_{\mathtt{East}}}(\mathtt{S4}_{\mathtt{North}})$
functions
CBQS
P^L
Output port
$\mathtt{S1}_{\mathtt{East}}$
In. port
$\mathtt{S4}_{\mathtt{North}}$
$\mathtt{S4}^{\dagger}_{\mathtt{North}}$
$\mathtt{S4}'_{\mathtt{North}}$
$\mathtt{S4}^{*}_{\mathtt{North}}$

Figure 3.14: Model for $\mathtt{S1}_{\mathtt{East}}$ when a regulator is placed for the parent $\mathtt{S4}_{\mathtt{North}}$

Theorem 3.2 (Fix-point result)
If the network with cyclic dependencies is empty at $t = 0$, then any non-negative fixed point, i.e., a vector $\overline{\boldsymbol{b}}$ such that $\mathcal{FF}(\overline{\boldsymbol{b}}) = \overline{\boldsymbol{b}}$, is a valid burst bound for the network with cyclic dependencies at the cuts. If $\mathcal{FF}$ has a finite non-negative fixed point, then the network is stable.

The proof is in Appendix B.1.2. In a recent work [Plassart, Le Boudec 2021], Plassart and Le Boudec propose several new approaches that do not require minimizing the number of cuts. In particular, Alternating TFA [Plassart, Le Boudec 2021, Algorithm 4] can be implemented with multi-threading. Plassart and Le Boudec also prove the equivalence of all approaches regarding their end-results: each approach, including the one published in [Thomas, Le Boudec, Mifdaoui 2019], computes the same delay bounds. A recent work from Anne Bouillard also proves that the fixed point is unique [Bouillard 2022].

3.3.4 Extension of the FP-TFA Approach to Networks With Regulators

If a vertex n has traffic regulators installed as optional functions, then we can distinguish its parents between the *regulated* parents (parent p for which a regulator $\mathrm{PFR}_n(p)$ or $\mathrm{IR}_n(p)$ is installed) and the *unregulated* parents for which no regulator is installed. We can then rewrite (3.1) into

$$\alpha_{n^\dagger} = \sum_{f|\mathrm{source}(f)=n} \alpha_{f,\phi} + \left(\sum_{\substack{p,\text{unregulated}\\ \text{parent of } n \text{ in } \mathcal{G}}} \alpha_{(p,n),n^\dagger} \right) + \left(\sum_{\substack{p,\text{regulated}\\ \text{parent of } n \text{ in } \mathcal{G}}} \alpha_{(p,n),n^\dagger} \right) \tag{3.9}$$

where the middle term is computed as previously and the last sum is simply the sum of the configured shaping curves for all the flows processed by the regulator $\mathrm{PFR}_n(p)$ or $\mathrm{IR}_n(p)$.

Example: Assume that we place a PFR that regulates the flows coming from $\mathtt{S4}_{\mathtt{North}}$ as in Figure 3.14. Then $\alpha_{(\mathtt{S4}_{\mathtt{North}},\mathtt{S1}_{\mathtt{East}}),\mathtt{S1}^{\dagger}_{\mathtt{East}}}$ simply equals the configuration of the PFR for g, *i.e.*,

$$\alpha_{(\mathtt{S4}_{\mathtt{North}},\mathtt{S1}_{\mathtt{East}}),\mathtt{S1}^{\dagger}_{\mathtt{East}}} = \gamma_{r_g,b_{g,\phi}}$$

This curve is shown in Figure 3.11b.

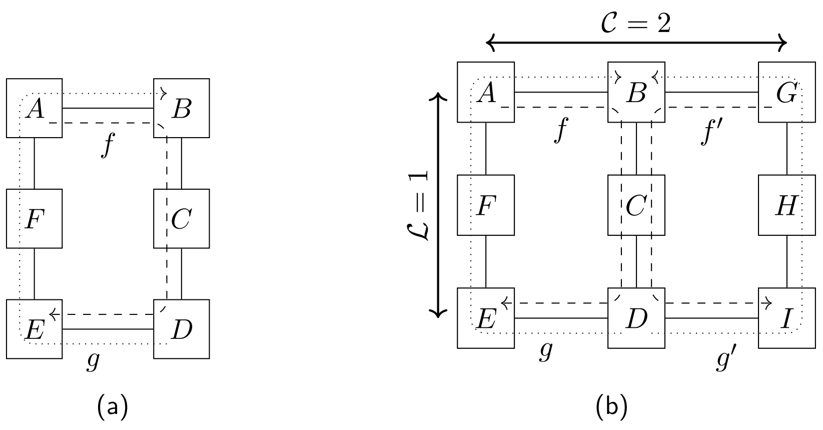


Figure 3.15: Parametric topology used for the evaluation (a): the basic cell, (b): a grid with two cells.

If the regulators have been placed using the LCAN tool, then the network is feed-forward.

3.4 Evaluation of the Partial-Deployment Approach

We now evaluate the performance of our partial deployment approaches (of either PFRs or IRs) in a synthetic use-case with respect to the three other approaches: no deployment of any regulator, full deployment of PFRs and full deployment of IRs. In the following, LCAN is configured with a fixed cost of 1 arbitrary unit for every regulator, irrespective of the number of flows that the regulator processes.

3.4.1 Network Description

We consider a basic network cell made of six switches with two flows, f and g (Figure 3.15a). By operating axial symmetries at the borders (axis (B, D) and axis (E, D)), we generate a grid made $\mathcal{L}$ lines and $\mathcal{C}$ columns of such basic cells. For example, Figure 3.15b corresponds to the grid $\mathcal{L} = 1$, $\mathcal{C} = 2$.

All the flows in the grid belong to the same class of interest and are constrained at their respective sources by the same leaky-bucket γ_{r,b_ϕ}. All flows have the same constant packet size $l = b_\phi$. The network is homogeneous and all ports provide the same service $\beta_{R,L}$, with R the rate and L the latency of the service. Furthermore, all transmission links have the same transmission capacity c.

We are interested in obtaining the worst end-to-end delay bound among all the flows of the network. Note that the path of any flow is always five-hops long, irrespective of the size of the network.

3.4.2 Parameters of Interest

In all the above parameters, we identify three values of interest:

- the network size: We want to observe how the LCAN approach scales to big topologies, and how many regulators it saves, with respect to the full deployment approach.

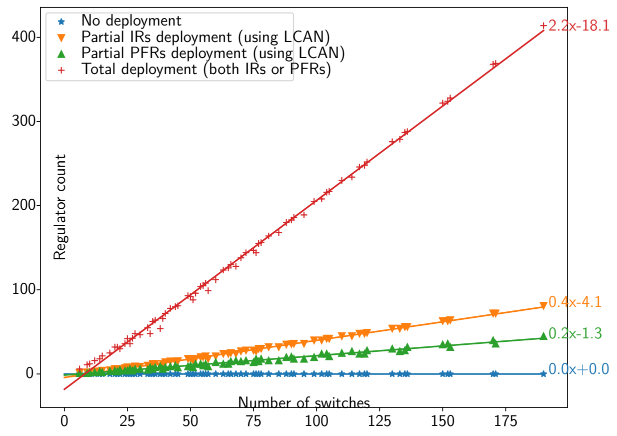


Figure 3.16: Number of regulators versus the grid network size for the different approaches.

- the network load u, equal to $u = 4r/R$ for $\mathcal{L} \geq 2$: We expect that the higher the network load is, the higher the delay bounds are. We seek to determine whether there exists a critical load u_{crit} beyond which FP-TFA cannot compute any delay bound in the no-deployment approach.
- the shaping factor R/c, with $R/c \leq 1$: We expect that the higher the shaping factor, the higher the line-shaping effect. We are interested in how the shaping factor affects the delay bounds and possibly the critical load u_{crit} for the no-deployment approach.

3.4.3 Effect of the Network Size on the Performance of LCAN

We first compute the number of required regulators in the different approaches for all grids $(\mathcal{L}, \mathcal{C}) \in [1, 9] \times [1, 9]$. Figure 3.16 shows the number of regulators as a function of the total number of switches in the grid. Each cross corresponds to a given $(\mathcal{L}, \mathcal{C})$ grid, and we highlight the best linear fit for each approach.

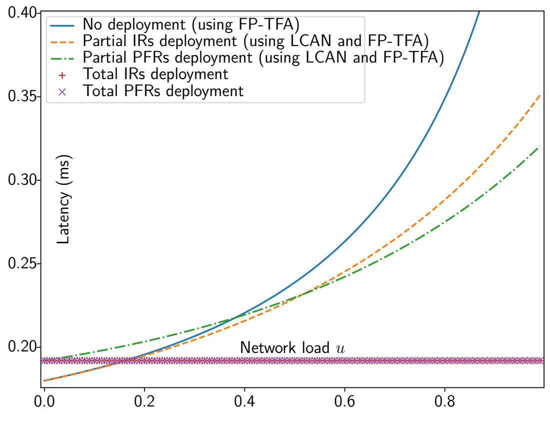
We note that the full-deployment approach requires more than two regulators per switch, whereas the partial-deployment approaches both require fewer than one regulator every two switches of the topology. Overall, the IR partial deployment requires 81% fewer regulators than the full-deployment approach and the gain increases up to 89% with PFRs.

This values are very encouraging because the grid used for the evaluation is a highly connected network that generates many cyclic dependencies. We do not expect the industrial networks to exhibit such complex patterns, and we can expect that the gain will be even higher for industrial networks.

3.4.4 Effect of the Network Load on the Latency Bounds

Figure 3.17a presents the delay bounds obtained with FP-TFA under the different approaches, as a function of the network load on the $(\mathcal{C} = 8, \mathcal{L} = 8)$ grid configuration (it corresponds to 153 switches). The delay bounds are obtained with a line shaping factor $R/c = 1$.

At low network load, the full deployment (of either PFRs or IRs), as well as the partial deployment of PFRs, perform worse than the no-deployment approach or the partial deploy-



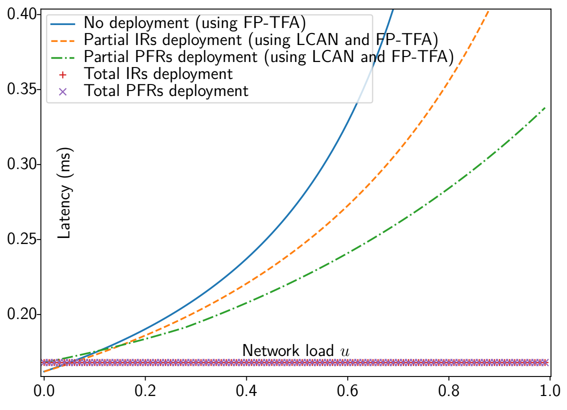


Figure 3.17: End-to-end latency bound of the flows as a function of the network load (lower is better). Size of the grid $\mathcal{L} = \mathcal{C} = 8$, initial burstiness $b_0 = 12\text{E}3\text{bits}$, service rate $R = 1\text{E}9\text{bits/s}$, service latency $T = 1.2\text{E}{-}5\text{s}$, packet length $l = 12\text{E}3\text{bits}$. (a) Shaping factor $R/c = 1$. (b) Shaping factor $R/c = 0.5$.

ment of IRs. Up to a threshold of $u \approx 15\%$, the best delay bounds are obtained without any regulator. On one hand, the traffic regulators have a negative effect on latency bounds, because they block the line-shaping effect (compare Figures 3.11a and 3.11b). On the other hand, the burstiness increase is limited at low load, hence the beneficial effect of the traffic regulators (blocking the burstiness increase) is limited and does not compensate for their above negative effects. This explains the full deployment approach that provides the larger delay bounds.

As the network load increases, however, the interest of regulators becomes noticeable. At high load, the best approaches are the full-deployment schemes, either with PFRs or with IR (both full-deployment approaches provide the same delay bounds). For the no-deployment approach, the values for the loads $u \geq 0.85$ are out of the scale in Figure 3.17a. However, FP-TFA was able to obtain latency bounds for all computation points, including 0.99, thus $u_{\text{crit}} \geq 0.99$ for this network.

At high load, the partial-deployment approach represents a good compromise, where it improves the delay bounds in comparison to the no-deployment approach but requires fewer regulators than the full-deployment approach that provides better bounds at high load.

3.4.5 Effect of the Line Shaping

In Figure 3.17b we diminishes the shaping factor R/c down to 0.5. This diminishes the line-shaping effect, *i.e.*, the beneficial effect on latency of the greedy-shaper modeling the transmission links (Section 2.6).

The value of the load threshold at which regulator deployments perform the best is reduced to $u \approx 5\%$. This is due to the combination of two effects: (1) All partial- and no-deployment approaches have a performance worse than with $R/c = 1$. (2) The delay bound of the full-deployment approach is decreased because it is only affected by the delay improvement from [Mohammadpour, Stai, Le Boudec 2019]. This improvement has a more positive effect when the transmission rate c is higher.

At $R/c = 0.5$, FP-TFA remains able to compute delay bounds for the entire spectrum of network loads ($u_{\text{crit}} \geq 0.99$) in the no-deployment approach. However, complementary tests show that, without regulators, the critical load of FP-TFA u_{crit} diminishes below 0.98 when the shaping factor diminishes below 0.33. This highlights that the line-shaping effect is decisive for keeping a high critical load u_{crit} despite the cyclic dependencies.

The approaches with partial deployment ensure feed-forward networks, hence their u_{crit} equals 1: the stability is guaranteed for any load $u < 1$ and any shaping factor R/c. However, the partial deployment of IRs shows very large bounds, even though LCAN places more IRs than PFRs. With fewer PFRs, we achieve a better performance. At a high utilization or for a high transmission rate, IRs need to be placed everywhere to provide noticeable improvements.

Conclusion

In this chapter we have discussed the issue of cyclic dependencies; they are a possible consequence of using multi-path topologies in time-sensitive networks.

In Section 3.1, we have discussed how the cyclic dependencies are managed in the literature. We have concluded that the cyclic dependencies were either: avoided, by using routing constraints; solved, by using specific approaches based on network calculus; or broken by deploying traffic regulators at every node in the network. We have observed that routing constraints are incompatible with redundancy requirements thus have decided not to consider them.

In Section 3.2, we have proposed a third approach, the *partial deployment* approach, that breaks all cyclic dependencies using as few regulators as possible. We developed the LCAN algorithm for breaking all cyclic dependencies at minimal cost by using either PFRs or IRs. Then in Section 3.3, we have described the FP-TFA algorithm, a fixed-point formulation of the TFA approach that can compute latency bounds in networks with cyclic dependencies and/or regulators. FP-TFA includes several improvements with respect to the classic TFA approach. In particular, it takes into account the line-shaping effect.

Finally in Section 3.4 we have evaluated the *partial-deployment* approach with LCAN with respect to the *full-deployment* and *no-deployment* approaches in a parametric topology. The latency bounds are computed by FP-TFA, even if the network contains cyclic dependencies. We have observed that the *partial-deployment* approach provides an interesting compromise between performance, stability, and the number of hardware elements. The partial deployment of IRs is an interesting option only at low utilization and transmission rates. When one of these values increases, this approach shows performance worse than partial deployment of PFRs, and IRs need to be placed everywhere to provide noticeable improvements.

The LCAN algorithm breaks all cyclic dependencies by using as few regulators as possible. However, it does not anticipate on the consequences of its choice on the latency bounds. A heuristic that anticipates on these aspects could be used for example to segregate between two solutions that have the same hardware cost. Between October 2019 and February 2020, Nicolò Dal Fabbro performed a semester project within the LCA2 laboratory at EPFL to investigate these aspects with the supervision of Prof. Le Boudec and advice from this thesis' author [Dal Fabbro 2020].

Multi-path topologies are even more likely to generate cyclic dependencies when redundant flows are mapped on these topologies. In the next chapter, we analyze the effect of redundancy mechanisms on latency bounds.

Chapter 4

Packet Replication and Elimination

"And you keep them running and in order ?"
"Right!"
"And if they break down?"
The tech-man shook his head indignantly, "They don't break down. They never break down. They were built for eternity."

Isaac Asimov, *The Foundation.*

Contents

4.1 Related Work on Packet Replication And Elimination 64

4.1.1 The Packet Replication and Elimination Functions in TSN and DetNet . 64

4.1.2 Configuration Options and their Analysis in the Literature 66

4.1.3 Main Challenges when Computing Delay Bounds in the General Case . . 66

4.2 Modeling the Redundancy Mechanisms 69

4.2.1 Changes to the Flow Model 69

4.2.2 Changes to the Device Model 71

4.2.3 Definition of the Worst-Case, Assumptions on Losses 74

4.3 Toolbox for the Deterministic Analysis of Packet Replication and Elimination Functions 75

4.3.1 Output Arrival Curve of a PEF 75

4.3.2 Reordering Introduced by the Packet Replication and Elimination 78

4.3.3 Obtaining End-to-End Delay Bounds in Networks with packet replication and elimination functions (PREFs) 81

4.4 Interactions between PREF and Traffic Regulators 82

4.4.1 Delay-Bound Analysis of PREFs Combined with Per-Flow Regulators . . 82

4.4.2 Instability of the Interleaved Regulator Placed after a Set of PEFs 84

4.4.3 Effect of the Packet-Ordering Function on the Combination of a PEF with Traffic Regulators 86

4.5 Evaluation of the Performance of The Toolbox 88

4.5.1 Description of the Evaluation 89

4.5.2 Comparison of the Delay Bounds when Varying the Network Size 89

4.5.3 Comparison of the Delay Bounds when Varying the Network Load 90

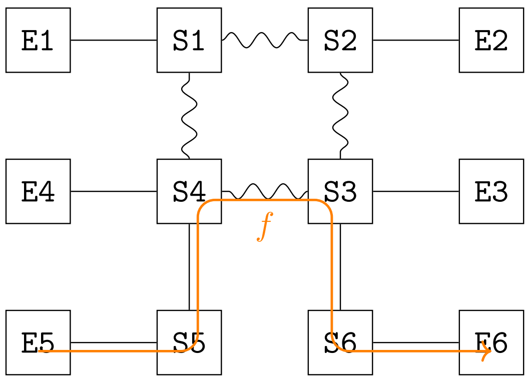


Figure 4.1: Physical topology used for the toy example throughout the chapter. It is obtained from the toy example of Chapter 3 by removing the switch S0 and adding the end systems E5, E6 and the switches S5, S6. In this chapter, we are interested in the flow f from E5 to E6.

In Chapter 3, we have discussed that the cyclic dependencies are possible consequences of using multi-path topologies and can have a significant effect on latency guarantees. In this chapter, we discuss how the redundancy mechanisms affect the latency guarantees of time-sensitive networks.

Indeed, time-sensitive networks provide a set of redundancy mechanisms that are of particular interest for safety-critical applications that require high levels of reliability. However, the use of the scheduling and redundancy mechanisms could concurrently affect the timing behavior of time-sensitive networks. Therefore, their efficiency and safety can be validated only through an appropriate worst-case timing analysis of both mechanisms.

In this chapter, we first discuss the related work on the analysis of redundancy mechanisms in Section 4.1. We then provide a framework for modeling redundancy mechanisms in the network-calculus theory through a model of the redundancy mechanisms in Section 4.2 and through a toolbox of network-calculus results in Section 4.3. In Section 4.4, we analyze the interactions between the traffic regulators (PFRs and IRs) and the redundancy mechanisms. In Section 4.5, we evaluate our framework on a parametric topology. Part of the material presented in this chapter was published in [Thomas, Mifdaoui, Le Boudec 2022].

4.1 Related Work on Packet Replication And Elimination

We first introduce the packet replication and elimination functions (PREFs) (redundancy mechanisms) as standardized in IEEE TSN and IETF DetNet. Then we discuss the related work on their analysis and highlight the main challenges posed by PREFs.

4.1.1 The Packet Replication and Elimination Functions in TSN and DetNet

The TSN *frame replication and elimination for redundancy* (FRER) for layer 2 and the DetNet *packet replication, elimination and ordering functions* (PREOFs) for layer 3 are seamless redundancy mechanisms that allow for redundant transmissions and for the elimination of duplicate packets.

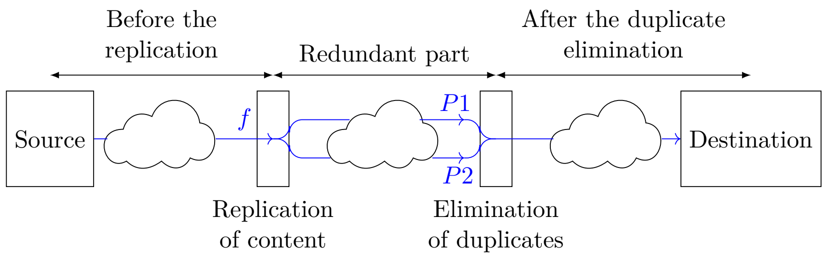


Figure 4.2: Path of a replicated flow f. It can be separated into three sections: before, in and after the redundant part.

Table 4.1: Main Acronyms Used in the Chapter and Comparison with the Terms of the Working Groups.

In this thesis		Term used in DetNet [RFC 8655]	Term used in TSN [21]
PREFs	Packet replication and elimination functions	Packet replication and elimination functions	**FRER:** Frame replication and elimination for redundancy [IEEE 802.1CB]
PRF	Packet replication function	Packet replication function	Stream splitting function [IEEE 802.1CB, §7.7]
PEF	Packet elimination function	Packet elimination function	Sequence recovery function [IEEE 802.1CB, §7.4.2]
POF	Packet ordering function	Packet ordering function	Does not exist in TSN (June 2022)
REG	Traffic regulator	Shapers [RFC 2475, §2.3.3]	**ATS:** Asynchronous traffic shaping [IEEE 802.1Qcr]

Example: Consider the toy example from the previous chapter to which we apply a few modifications. We first remove the bridge `S0` and we add two bridges `S5`, `S6` and two end-stations `E5` and `E6`. This gives the physical topology shown in Figure 4.1.

Consider a flow f of safety-critical data from `E5` to `E6` that we first route through the shortest path (Figure 4.1). But further assume that the flow's end-to-end packet loss requirement is so stringent that the backbone links (wavy lines in Figure 4.1) are not reliable enough to meet this requirement: the backbone links may lose some packets, too often with respect to the flow's end-to-end loss-ratio requirement.

Redundancy mechanisms such as TSN FRER and DetNet PREOFs can be used to decrease the end-to-end packet loss ratio by distributing "*the contents of [...] flows over multiple paths in time and/or space, so that the loss of some of the paths does need not cause the loss of any packets*" [RFC 8655]. To do so, the DetNet packet-replication function (PRF) replicates each incoming packet into several outgoing packets that can take different paths (Figure 4.2). The paths then merge and multiple copies of the packet (the replicates) reach a packet-elimination function (PEF) that forwards only the first replicate and eliminates the subsequent ones (the duplicates). The PEF generally relies on a sequence number in the packet header to identify the replicates [RFC 8655]. In Table 4.1, the different acronyms used in this thesis are compared to the names of the mechanisms in the TSN and the DetNet working groups.

[RFC 8655, §3.1] recalls that the use of packet replication and elimination functions (PREFs) is "*constrained by the need to meet the users' latency requirements*". Therefore, understanding how PREFs affect the worst-case latency guarantees is fundamental to: (i) determine the applicability of PREFs in industrial networks; (ii) perform trade-offs between

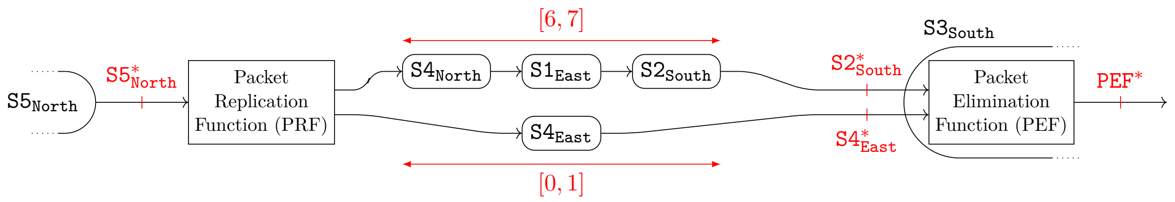


Figure 4.3: An excerpt of the flow graph $\mathcal{G}(f)$ for the toy example used throughout the chapter. The flow f is replicated and sent through two paths, $(\texttt{S4}_{\texttt{North}}, \texttt{S1}_{\texttt{East}}, \texttt{S2}_{\texttt{South}})$ and $(\texttt{S4}_{\texttt{East}})$, with different delay bounds. The paths then merge into F, that removes the duplicates.

latency and loss-ratio requirements; and (iii) design networks with stringent requirements on both aspects.

4.1.2 Configuration Options and their Analysis in the Literature

PREFs can be used with various configuration options. For example [IEEE 802.1CB, Annex B] states that TSN FRER is inter-operable with High-availability Seamless Redundancy (HSR) and Parallel Redundancy Protocol (PRP), two redundancy mechanisms standardized in [IEC 62439-3] and used in automotive and industrial applications.

Table 4.2 summarizes the various configuration options when using PREFs for the flow in Figure 4.1, as well as their analysis in the literature. The first two rows of Table 4.2 are the only configurations on which a deterministic analysis of the latency bounds was conducted using network calculus. These analyses are limited to the assumption of using redundancy mechanisms at the end systems. The last row of Table 4.2 has only been studied using simulations, that do not provide latency guarantees.

In this chapter, our primary goal is to bridge these gaps and to provide a method of worst-case timing analysis for time-sensitive networks that implement redundancy mechanisms in the general use-case, *i.e.*, at end systems and/or intermediate nodes.

4.1.3 Main Challenges when Computing Delay Bounds in the General Case

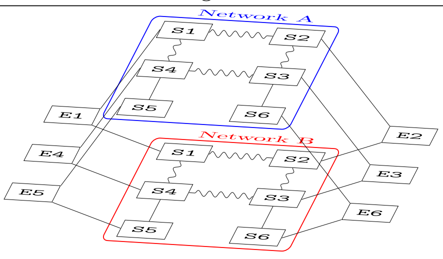
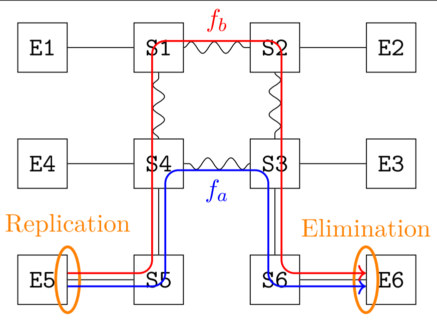
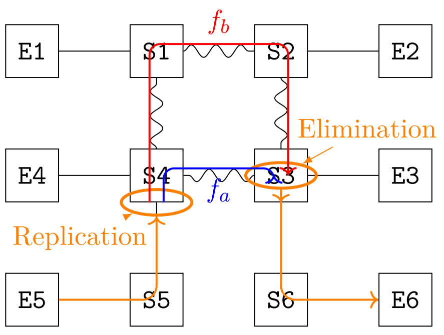
Two main challenges arise when analyzing the general case. First, the traffic exiting the PEF can exhibit both an increased burstiness and a mis-ordering of the packets. This can lead to increased delay bounds in the nodes placed after the PEF. Second, the coexistence of the packet mis-ordering with the burstiness increase could negatively affect the behavior of the devices that have been designed to tackle each issue individually.

Increased Burstiness and Mis-Ordering

Example: Consider the PREFs configuration of the last row of Table 4.2. The graph of the flow f is shown in Figure 4.3. Assume that the flow has the leaky-bucket arrival curve $\alpha_{f,\texttt{S5}^*_{\texttt{North}}} = \gamma_{r_0,b_0}$ at the output of $\texttt{S5}_{\texttt{North}}$, with a rate r_0 of one data unit per time unit (d.u./t.u.) a burst $b_0 = 1$d.u. Further assume that the flow suffers along each sub-path a delay bounded by $[6, 7]$ t.u. (resp., $[0, 1]$ t.u.), as shown in Figure 4.3.

An acceptable trajectory is given in Figure 4.4. Here, the path through $\texttt{S4}_{\texttt{East}}$ drops all data units from 1 to 6: they are only received through the longer path with a latency

Table 4.2: Overview of the Configuration Options supported by TSN FRER and DetNet PREOFs and their Analysis in the Literature.

Configuration	Related work
 Network A S1 S2 S3 S4 S5 S6 Network B S1 S2 S3 S4 S5 S6 E1 E2 E3 E4 E5 E6 Use of two parallel, identical networks. Each end-systems is connected to both. Examples: • [AFDX] in Airbus A380, A350, A400M. • Parallel Redundancy Protocol (PRP) [IEC 62439-3].	Each sub-network is independant and can be analyzed with network calculus. The overall worst-case latency is the maximum of the worst-case latencies on each sub-network. [Li, *et al.* 2017] analyze the possibility that packets could be delivered to the destination out-of-order, which AFDX end-systems do not tolerate. They propose means to mitigate the issue.
 f_b E1 S1 S2 E2 E4 S4 S3 E3 f_a Replication Elimination E5 S5 S6 E6 Sub-flows with replication at the source, elimination at the destination. Examples: • High-availability Seamless Redundancy (HSR) [IEC 62439-3] for ring topologies.	With respect to Figure 4.1, a new flow f_b is created and increases the delay and backlog bounds in the crossed nodes. It is even susceptible to generate cyclic dependencies (see Section 3.1.3). Yet both sub-flows can be considered to be two independent flows and the techniques developed in Chapter 3 can be used to manage the induced cyclic dependencies. In [Heise, *et al.* 2014], a modified AFDX network that follows the HSR principles is analyzed. The authors highlight the issue of cyclic dependencies. They apply an ad-hoc version of the fixed-point (Theorem 3.2) to obtain delay bounds for each redundant flow and simply take the maximum of the two obtained latency bounds for the overall latency.
 f_b E1 S1 S2 E2 Elimination E4 S4 S3 E3 f_a Replication E5 S5 S6 E6 Sub-flows with replication and elimination at intermediate nodes.	In [Heise, Geyer, Obermaisser 2016], a simulation framework based on OMNeT++ has been developed for TSN mechanisms, including TSN frame replication and elimination for redundancy (FRER) [Heise 2018, §4.3.2]. In [Pahlevan, Obermaisser 2018], simulation models for the Riverbed simulator are proposed. In [Hofmann, Nikolić, Ernst 2020] and [IEEE 802.1CB, §C.9], several concerns about using TSN FRER in this configuration have been discussed, including the issue of obtaining latency bounds when elimination is performed at intermediate node. To the best of our knowledge, our work [Thomas, Mifdaoui, Le Boudec 2022] is the first to provide latency bounds in this configuration.

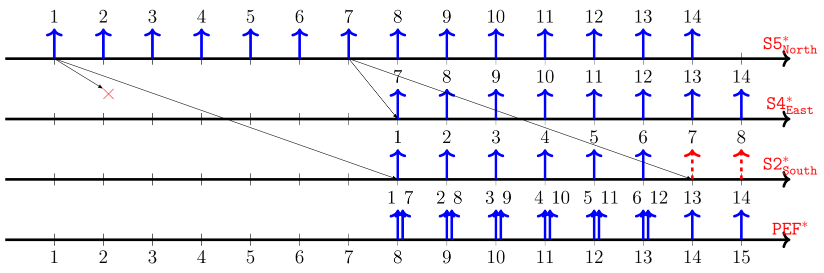


Figure 4.4: An example trajectory on the toy example of Figure 4.3 that causes the output traffic after the PEF to exhibit a double rate for some period of time.

of 7 t.u. After, the link through $\texttt{S4}_{\texttt{East}}$ becomes available and the data units 7 to 14 are received through both paths, with a latency of 1 t.u. [resp., 7 t.u.]. The PEF receives the sum of $\texttt{S2}^*_{\texttt{South}}$ and $\texttt{S4}^*_{\texttt{East}}$. It drops the duplicates and forwards the packets that contain never-observed data units. Its output is shown on the line $\texttt{PEF}^*$.

We observe that the traffic after the PEF is much more bursty than before the replication function: between 8 t.u. and 13 t.u., the output of the PEF shows a doubled rate, $2r_0$.

The example suggests that packet replication and elimination functions (PREFs) can significantly increase the flows' burstiness, which could further worsen the congestion and the worst-case delay in the downstream nodes. A first approach for bounding the traffic of the flow after the PEF, which we denote as *intuitive*, consists in doing as if the PEF would never drop a packet (i.e., even the duplicates are forwarded). This approach requires the network engineer to dimension all the downstream nodes in order to support a sustained double rate. It leads to loose end-to-end latency bounds, as we show in Section 4.5.

Figure 4.4 also highlights that PREFs can create a mis-ordering of the data uits: d.u. 6 exits the PEF 5 t.u. after d.u. 7. Obtaining an upper bound for this mis-ordering is important for comparing it to the application's requirements. We provide such bound in Theorem 4.2.

Interactions with Packet Ordering Functions and Traffic Regulators

If the receiving application does not tolerate any mis-ordering, then the DetNet packet-ordering function (POF) [RFC 8655, §3.2.2.2] can be used after the PEF to correct the mis-ordering introduced by PREFs. Similarly, if the end-to-end latency of a flow does not meet its requirements due to a high worst-case delay after the PEF (third section of Figure 4.2), then the traffic regulators introduced in Section 3.1.5 can be used for removing the burstiness increase caused by PREFs thus reducing the worst-case delay in downstream nodes.

However, as mentionend in Section 3.1.5, many properties of the regulators rely on the assumption that the upstream system is FIFO. As observed on the toy example, this assumption does not hold with PREFs.

Example: Assume that the traffic regulator in Figure 4.5 shapes the traffic back to the profile it had at the input $\texttt{S5}^*_{\texttt{North}}$. In terms of burstiness, this makes the middle section in Figure 4.2 transparent to the third section. The regulator processes the traffic from the $\texttt{PEF}^*$ line of Figure 4.4 and forces the packets to be as spaced as in the $\texttt{S5}^*_{\texttt{North}}$ line by delaying and storing the packets if required. Clearly, the upstream system between

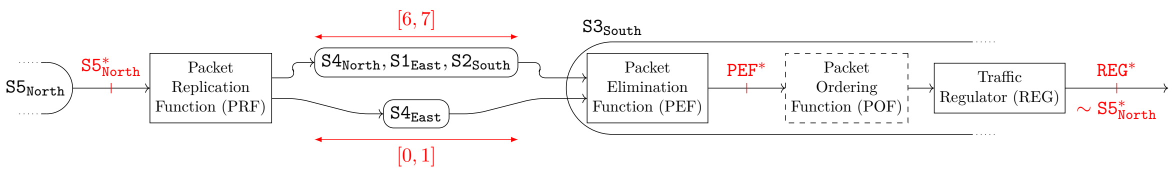


Figure 4.5: The toy example of Figure 4.3, extended with a POF and a REG to deal with the mis-ordering and burstiness increase issues due to PEF.

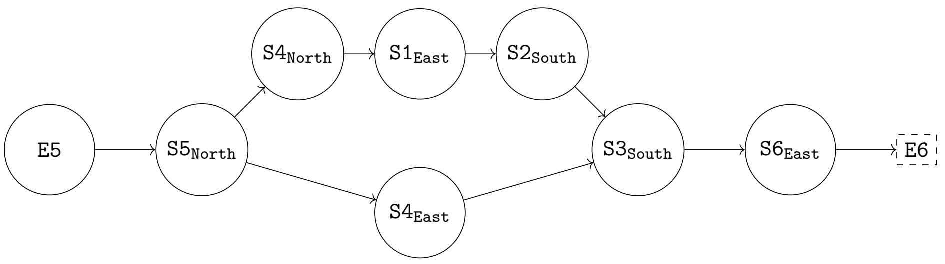


Figure 4.6: The entire flow-graph $\mathcal{G}(f)$ of the flow f in the toy example, corresponding to the last row of Table 4.2. The destination E6 (dashed rectangle) is shown for convenience but is not present in the actual graph.

$\texttt{S5}^*_{\texttt{North}}$ and PEF* in Figure 4.5 is not FIFO, because the packets exit the PEF out of order (Figure 4.4). Thus the properties of the regulators that depend on this assumption might not hold and the cohabitation of the PEF and the regulator (REG) could negatively affect the latency bounds. A packet-ordering function (POF) (dashed box in Figure 4.5) can be used after the PEF and before the regulator to force the upstream system to be FIFO. If such POF is placed, then we would expect to retrieve all the properties of the regulators.

In Section 4.4, we analyze the interactions between PREFs, POFs and regulators.

4.2 Modeling the Redundancy Mechanisms

Before detailing our network-calculus toolbox of results for solving the challenges described in Section 4.1.3, we discuss herein how the network-calculus model of Chapter 2, Section 2.6 is adapted to include packet replication and elimination functions (PREFs).

4.2.1 Changes to the Flow Model

Flow Graphs

The flow graph $\mathcal{G}(f)$ of a flow f continues to be defined as per Definition 2.9, but $\mathcal{G}(f)$ is no longer assumed to be a multicast tree: its sub-paths can also merge. $\mathcal{G}(f)$ is simply assumed to be a directed acyclic sub-graph of the GIF with a unique root, the source of f.

Example: The graph of flow f is shown in Figure 4.6. This is not a tree but it remains acyclic and contains a unique root, E5, the source of f.

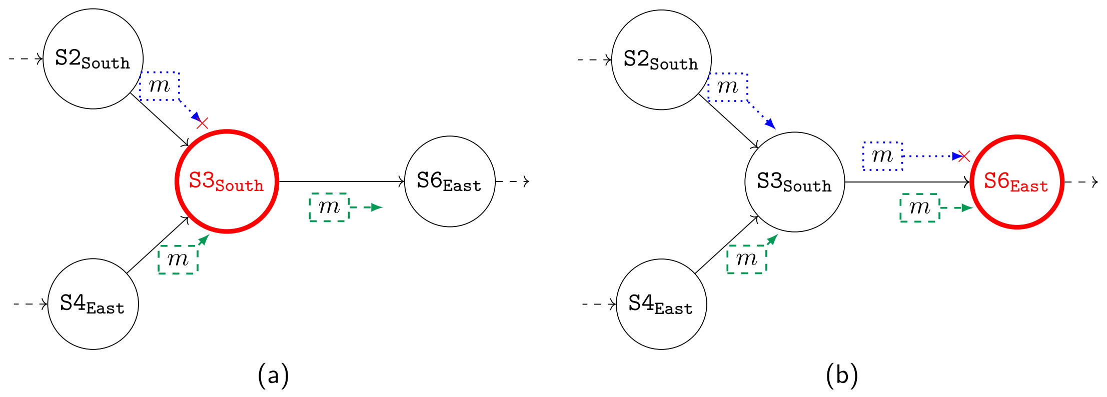


Figure 4.7: Illustration of different positions for the PEF in the toy example. S3$_{\texttt{South}}$ might receive the data unit m twice (in the dashed green and the dotted blue packets). Here we assume that the green packet is received prior to the blue packet. (a) S3$_{\texttt{South}}$ contains a PEF (thick red vertex), it drops the dotted blue packet that contains the already-seen data unit m (the blue packet is a duplicate for S3$_{\texttt{South}}$). (b) S3$_{\texttt{South}}$ does not contain any PEF for f, it forwards both replicates. S6$_{\texttt{East}}$ contains a PEF (thick red vertex), it receives the data unit twice and drops the dotted blue packet.

Data Unit versus Packet and Replicate versus Duplicate

We continue to use the distinction between data units (the abstract content) and packets (the physical container) as described in Section 2.6.3: at any given point in time, a data unit can be located at several locations in the network, transported by different packets. When the path diverges, the switching fabric plays the role of the packet-replication function (PRF) and we call replicates the set of packets that transport the same data unit.

When the sub-paths merge, an observer located after the merge point could observe the same data unit several times, in different packets. We say that a packet is a duplicate *for* an observation point [resp., *for* a function] if another replicate of the same data unit was already observed at this location [resp., by this function] in the past.

Additionally, if a packet is lost at some point, it doesn't mean that the data unit is lost for all downstream nodes: downstream observers could observe the data unit coming from another path. The notion of lost data unit is here again relative to an observer:

Definition 4.1 (Lost data unit for an observer) *For a data unit m that belongs to a flow f and for an observation point w or a function* **FUN** *that belongs to one of the vertices of $\mathcal{G}(f)$, we say that the data unit m is* lost *for w [resp., for* **FUN**] *if w [resp.,* **FUN**] *never observes m in any packet.*

Example: In Figure 4.6, if vertex S1$_{\texttt{East}}$ fails, then all the data units contained in the lost packets are *lost for* vertex S2$_{\texttt{South}}$, but not necessarily for vertex S6$_{\texttt{East}}$.

Position of PRFs and PEFs in a Flow Graph

When a node n has several children in a flow graph $\mathcal{G}(f)$, then the switching fabric acts as a packet-replication function (PRF). When a vertex, such as S3$_{\texttt{South}}$ in Figure 4.6, has several parents in $\mathcal{G}(f)$, it can receive the same data unit several times, within different packets.

However, it does not necessarily implement a PEF: the elimination of the duplicates can be deferred to a latter vertex.

If a PEF is present on such a vertex (case of $\texttt{S3}_{\texttt{South}}$ in Fig. 4.7a), then it forwards only the first replicate of the data unit. Any subsequent replicate (*i.e.*, any duplicate *for* $\texttt{S3}_{\texttt{South}}$) is dropped. If the vertex does not contain a PEF (case of $\texttt{S3}_{\texttt{South}}$ in Fig. 4.7b), then it forwards all the packets, and might consequently forward the same data unit several times.

Elimination-Pending Vertices

The route of a flow is defined by its flow graph $\mathcal{G}(f)$ and by the knowledge of the vertices that implement the PEF. In the figures, such vertices are marked with a thick red outline.

The vertices that are between a merge point and the nearest downstream PEF are the only vertices that can observe a data unit several times, we call them EP-vertices.

Definition 4.2 (EP-vertex) *For a flow f, the vertices of $\mathcal{G}(f)$ that are downstream of a vertex with several parents (included) and upstream of a the next PEF for f (excluded) are called* elimination-pending *vertices (EP-vertices) of $\mathcal{G}(f)$.*

Example: In Figure 4.7b, $\texttt{S3}_{\texttt{South}}$ is the only EP-vertex of $\mathcal{G}(f)$. Figure 4.7a does not contain any elimination-pending (EP)-vertex.

Diamond Ancestor

For n a vertex of $\mathcal{G}(f)$, there exists a path $(n_0 \rightarrow \cdots \rightarrow n)$ in $\mathcal{G}(f)$ that reaches n from n_0, the source of f. However, as $\mathcal{G}(f)$ is not a tree, such a path is not necessarily unique.

Example: In Figure 4.6, $\texttt{S3}_{\texttt{South}}$ is a descendant of $\texttt{E5}$ and two paths reach $\texttt{S3}_{\texttt{South}}$ from $\texttt{E5}$. We note that $\texttt{S5}_{\texttt{North}}$ plays a special role with respect to $\texttt{S3}_{\texttt{South}}$ in the graph $\mathcal{G}(f)$: it belongs to all the paths that reach $\texttt{S3}_{\texttt{South}}$ from the root. We say that $\texttt{S5}_{\texttt{North}}$ is a diamond ancestor of $\texttt{S3}_{\texttt{South}}$.

Definition 4.3 (Diamond ancestor) *For two vertices a and n in a flow graph $\mathcal{G}(f)$, we say that a is a diamond ancestor of n in $\mathcal{G}(f)$ if:*
– a is not an EP-vertex of $\mathcal{G}(f)$, and
– all paths in $\mathcal{G}(f)$ from the graph root to n contain a.

Example: Assume that the elimination function is placed in $\texttt{S6}_{\texttt{East}}$, as in Figure 4.7b. Then $\texttt{S5}_{\texttt{North}}$ and $\texttt{E5}$ in Figure 4.3 are diamond ancestors of $\texttt{S6}_{\texttt{East}}$ in $\mathcal{G}(f)$. However, $\texttt{S3}_{\texttt{South}}$ is not a diamond ancestor of $\texttt{S6}_{\texttt{East}}$ in $\mathcal{G}(f)$ because $\texttt{S3}_{\texttt{South}}$ is an EP vertex of $\mathcal{G}(f)$.

4.2.2 Changes to the Device Model

Like traffic regulators, the PEF and the POF are optional functions located before the CBQS (Figure A.1 of the *vade mecum* Appendix A). In the following, consider a vertex n of the graph induced by flows (GIF) $\mathcal{G}$.

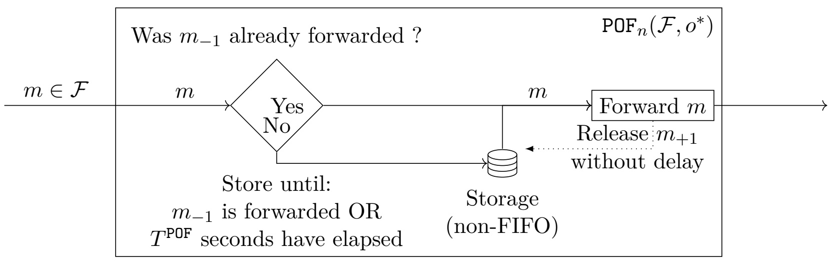

Was m_{-1} already forwarded ?
$\mathtt{POF}_n(\mathcal{F}, o^*)$
$m \in \mathcal{F}$
m
Yes
No
m
Forward m
Release m_{+1} without delay
Store until: m_{-1} is forwarded OR $T^{\mathtt{POF}}$ seconds have elapsed
Storage (non-FIFO)

Figure 4.8: Functional model of the packet-ordering function $\mathtt{POF}_n(\mathcal{F}, o^*)$. For a data unit m, m_{-1} [resp., m_{+1}] refers to the data unit of the aggregate $\mathcal{F}$ that crossed o^* just before [resp., just after] m.

The packet-elimination function (PEF)

For a flow f crossing n, the output port in n can contain a *packet-elimination function* (PEF) for flow f, noted $\mathtt{PEF}_n(f)$. For each incoming packet of f, $\mathtt{PEF}_n(f)$ determines if it has already observed the data unit contained in the packet. If so, the packet is identified as a duplicate *for* the PEF and is discarded. For the stream of packets that contains never-seen data units of f, the $\mathtt{PEF}_n(f)$ is transparent: FIFO and without any delay.

The packet-ordering function (POF)

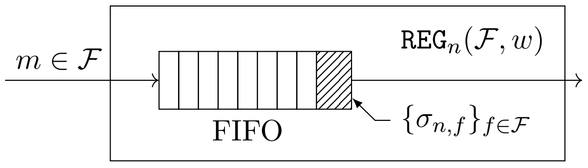
Consider a set of flows $\mathcal{F}$ crossing n and a vertex o such that for each flow $f \in \mathcal{F}$, o is a diamond ancestor of n in $\mathcal{G}(f)$. The output port in n can contain a *packet-ordering function* (POF) for the aggregate $\mathcal{F}$ with reference o^* (the output of vertex o), noted $\mathtt{POF}_n(\mathcal{F}, o^*)$. We assume that $\mathtt{POF}_n(\mathcal{F}, o^*)$ has the knowledge of the order in which the data units belonging to the aggregate $\mathcal{F}$ crossed the observation point o^*. $\mathtt{POF}_n(\mathcal{F}, o^*)$ then enforces the same order at its own output, by delaying the packets that are out of order.

However, a data unit m cannot be delayed by $\mathtt{POF}_n(\mathcal{F}, o^*)$ for a duration longer than the POF's timeout parameter $T^{\mathtt{POF}}$: After being stored for a duration $T^{\mathtt{POF}}$, m is immediately released, even if the previously-expected data unit has not been received so far. The timeout allows the POF to recover from losses without blocking the following data units forever [Mohammadpour, Le Boudec 2021; Varga, *et al.* 2021]. We assume that the timeout value of every POF conforms with the recommendations of [Mohammadpour, Le Boudec 2021, §IV.B]: It can only be triggered when one of the data units m of $\mathcal{F}$ is lost for the POF.

A POF cannot be placed at an EP-vertex: we always assume that the duplicates are eliminated before the flow is handed to the POF, which is consistent with the assumptions in [Varga, *et al.* 2021, §4.1]. The model of POF is illustrated in Figure 4.8. A possible implementation is given in [Mohammadpour, Le Boudec 2021, §3.4] and [Varga, *et al.* 2021].

The regulator (REG)

In the precedent chapter (Chapter 3), we have used the notation $\mathtt{PFR}_n(p)$ and $\mathtt{IR}_n(p)$ to denote a PFR [resp., an IR] configured by the LCAN algorithm and installed on vertex n to process all the flows coming from the parent p. In this chapter, we slightly modify the notation to make it more flexible in the context of PREFs.



$m \in \mathcal{F}$
$\texttt{REG}_n(\mathcal{F}, w)$
FIFO
$\{\sigma_{n,f}\}_{f\in\mathcal{F}}$

Figure 4.9: Model of a regulator $\texttt{REG}_n(\mathcal{F}, w)$, with shaping curves $\{\sigma_{n,f}\}_f$. The regulator only looks at the head-of-line packet.

Consider a set of flows $\mathcal{F}$ crossing n, and consider a vertex o of $\mathcal{G}$ such that, for each flow $f \in \mathcal{F}$, o is a diamond ancestor of n in $\mathcal{G}(f)$. Consider finally an observation point, $w \in \{o^\dagger, o', o^*\}$ (Figure A.2). The output port in n can contain a *regulator (REG)* for the aggregate $\mathcal{F}$ with reference w, noted $\texttt{REG}_n(\mathcal{F}, w)$. As in Chapter 3, a regulator is FIFO for $\mathcal{F}$ and is configured with a set of shaping curves, one per flow f of the aggregate $\mathcal{F}$, that we note $\{\sigma_{n,f}\}_{f\in\mathcal{F}}$.

In the context of PREFs, the packets of any flow f in the aggregate $\mathcal{F}$ can enter n through several input ports (n can have several parents in $\mathcal{G}(f)$). Thus a regulator can be limited to the packets of the aggregate that enter n coming from a specific parent p of n. We note $\texttt{REG}_n(\mathcal{F}, w, p)$ the regulator that processes only the stream of packets of the aggregate $\mathcal{F}$ that enter n from the edge $p \to n$, where, for each $f \in \mathcal{F}$, p is a parent vertex of n in $\mathcal{G}(f)$.

In Chapter 3, a per-flow regulator $\texttt{PFR}_n(p)$ configured by LCAN is noted

$$\texttt{PFR}_n(p) = \{\texttt{REG}_n(\{f\}, \phi, p)\}_{f\in\{h|h\ni(p,n)\}}$$

$\texttt{PFR}_n(p)$ a set of parallel elementary REGs, each processing a unique flow f of the aggregate by enforcing the source arrival curve ($w = \phi$) and by processing only the packets of f that comes from the parent p, which was clear in Chapter 3 because n had at most one parent in $\mathcal{G}(f)$.

Similarly, an interleaved regulator $\texttt{IR}_n(p)$ of Chapter 3, configured by LCAN is noted

$$\texttt{IR}_n(p) = \texttt{REG}_n(\{f|f \ni (p,n)\}, p^\dagger, p)$$

The IR is a unique elementary REG that processes the aggregate $\{f|f \ni (p,n)\}$ and enforces for each flow f of this aggregate the arrival curve that the flow had just before the CBQS of the parent p, *i.e.*, at $p^\dagger$ (see Figure A.2). Here again only the packets coming from p are processed, which, in the previous chapter, was equivalent to the aggregate $\{f|f \ni (p,n)\}$.

Order of the Optional Functions in a Vertex

We assume that each flow f can only be processed by at most one function of each type in each vertex. As in the technical documents [Varga, *et al.* 2021, §4.1], we also assume that, within a vertex, POF comes always after PEF.

Regulators (REGs) and PREFs are defined in two separate documents in IEEE TSN [IEEE 802.1CB; IEEE 802.1Qcr]. Hence, their exists an uncertainty on their relative order. The uncertainty has not been clarified as of June 2022. In the thesis, and as intuited in Section 4.1.3, we see the traffic regulators as of particular interest when they are placed after the PEFs, because they can shape the traffic back to the profile it had at the input of the

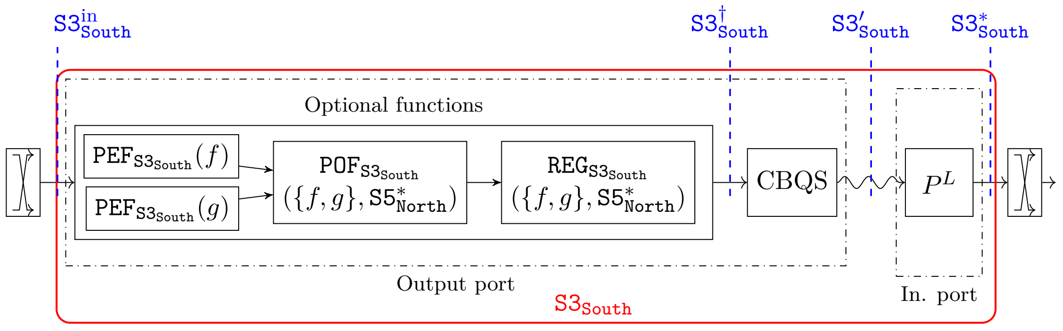

$\mathtt{S3}^{\text{in}}_{\mathtt{South}}$
$\mathtt{S3}^{\dagger}_{\mathtt{South}}$
$\mathtt{S3}'_{\mathtt{South}}$
$\mathtt{S3}^{*}_{\mathtt{South}}$
Optional functions
$\mathtt{PEF}_{\mathtt{S3}_{\mathtt{South}}}(f)$
$\mathtt{PEF}_{\mathtt{S3}_{\mathtt{South}}}(g)$
$\mathtt{POF}_{\mathtt{S3}_{\mathtt{South}}}(\{f,g\},\mathtt{S5}^{*}_{\mathtt{North}})$
$\mathtt{REG}_{\mathtt{S3}_{\mathtt{South}}}(\{f,g\},\mathtt{S5}^{*}_{\mathtt{North}})$
CBQS
P^L
Output port
$\mathtt{S3}_{\mathtt{South}}$
In. port

Figure 4.10: Example of an organization of the optional functions within an output port. After their respective PEF, the two flows share the same POF and the same regulator (REG). Note that a packet of f might be delayed by packet of g in both the POF and the REG, even if the order of its own packets is maintained and even if it already complies with the shaping curve $\sigma_{n,f}$ within the REG.

redundant section (second section in Figure 4.2). Analyzing the interactions between PREFs and REGs in this configuration is one of our major objectives, while placing a regulator before the PEF is equivalent to shaping the traffic within a sub-path of a multi-cast flow, a situation that can be analyzed using the results of Chapter 3.

To summarize, we assume that each output port can contain the following optional functions, in this order: PEFs → POFs → REGs.

Example: Consider the flow f in Figure 4.6 and assume the existence of a second flow, g that has exactly the same flow graph.

The output port $\mathtt{S3}_{\mathtt{South}}$ processes streams of packets coming from both parents $\mathtt{S4}_{\mathtt{East}}$ and $\mathtt{S2}_{\mathtt{South}}$. A first possible example of the organization of the functions before the CBQS within vertex $\mathtt{S3}_{\mathtt{South}}$ is shown in Figure 4.10. Each flow is first processed by its respective PEF, then both duplicate-free flows are reordered as an aggregate by using $\mathtt{POF}_{\mathtt{S3}_{\mathtt{South}}}(\{f,g\},\mathtt{S5}^{*}_{\mathtt{North}})$. This function enforces the same order for the aggregate $\{f,g\}$ as the one at the output of $\mathtt{S5}_{\mathtt{North}}$, *i.e.*, before the redundant section. Last, they are both processed by the same interleaved regulator that enforces two different contracts for f and for g, but keeps the aggregate $\{f,g\}$ FIFO. The two shaping curves $\sigma_{\mathtt{S3}_{\mathtt{South}},f}$ and $\sigma_{\mathtt{S3}_{\mathtt{South}},g}$ can differ, but each must be an arrival curve of its respective flow at $\mathtt{S5}^{*}_{\mathtt{North}}$.

A variant of this situation is shown in Figure 4.11. After elimination, each flow is now independent from the other one, where the POFs enforce per-flow order and the two REGs are per-flow regulators (PFRs). This situation is different from Figure 4.10 because a packet of f cannot be delayed by a packet of g. In addition it could have a higher cost.

4.2.3 Definition of the Worst-Case, Assumptions on Losses

With the exception of PEF, each function, each CBQS, each switching fabric, each input port and each internal connection within a device is assumed to be lossless (does not lose any packets). Packets can be lost on the transmission links between devices. This model covers various failures, including random media losses, the shutdown of an output port (equivalent to its out-going link losing all packets) and the shutdown of an input port (equivalent to its

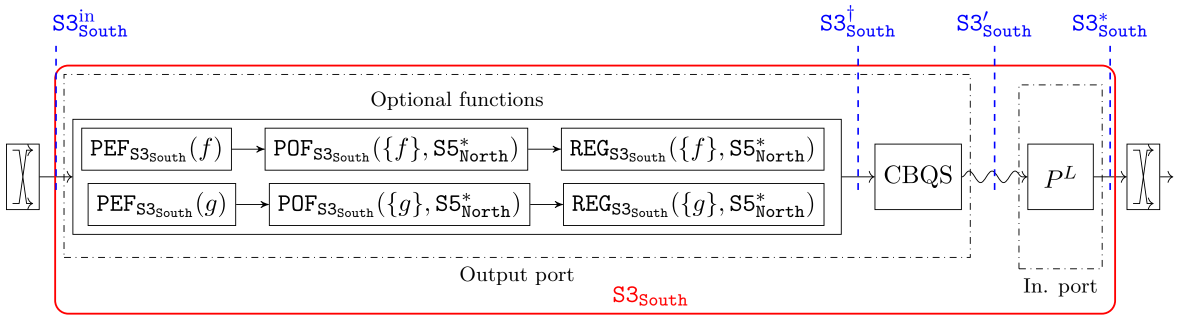


Figure 4.11: Another example of the organization of the optional functions within an output port with one POF and one REG per flow. From the hardware perspective, this configuration could have a higher cost than in Figure 4.10. Note that every flow is independent from the other one, before reaching the CBQS.

in-going link losing all packets).

Definition 4.4 (Delay bounds for a flow f) *Let f be a flow and d one of the destinations of f. An end-to-end (ETE) delay upper bound [resp., lower bound] of f for d is an upper bound [resp., lower bound] on the maximum [resp., minimum] delay that each data unit m of f takes to reach d, assuming that the data unit m is not lost for d.*

As packets can be lost on transmission links, the network is not assumed to be lossless. Of course, the latency bounds computed in this paper are only valid for the non-lost data units (the data units for which at least one replicate reaches the destination), but these bounds remain valid even if some other data units are lost in the network.

4.3 Toolbox for the Deterministic Analysis of Packet Replication and Elimination Functions

We provide here our toolbox of results for modeling packet replication and elimination functions (PREFs) in the network-calculus framework. We first compute a tight arrival-curve characterization of the traffic at the output of a PEF. We then quantify the amount of re-ordering that PREFs can cause and analyze the consequences of correcting this mis-ordering by using a packet-ordering function (POF).

4.3.1 Output Arrival Curve of a PEF

Theorem 4.1 (Output arrival curve of a PEF)
Let $PEF_n(f)$ be a packet-elimination function for flow f at vertex $n \in vertices(\mathcal{G}(f))$. Assume that $\alpha_{f,PEF^{in}}$ is an arrival curve of f at the input of $PEF_n(f)$. Then

1/ $\alpha_{f,PEF^{in}}$ is an arrival curve for the flow at the output of the PEF.

2/ For every diamond ancestor a of n in $\mathcal{G}(f)$, assume that α_{f,a^*} is an arrival curve for f at the output of a and denote by $d_f^{a^*\to n^{in}}$ [resp., $D_f^{a^*\to n^{in}}$] a minimum [resp., maximum] delay bound for f between the output of a and the input of $PEF_n(f)$ (i.e., *of n), along any possible paths $a \to n$ within the graph $\mathcal{G}(f)$. Then*

$$\alpha_f^{a\to n} \triangleq \alpha_{f,a^*} \oslash \delta_{(D_f^{a^*\to n^{in}})-(d_f^{a^*\to n^{in}})} \tag{4.1}$$

is an arrival curve for f at the output of the PEF.

Furthermore, the min-plus convolution of all above arrival curves

$$\alpha_{f,PEF^*} = \alpha_{f,PEF^{in}} \otimes \alpha_f^{a_1\to n} \otimes \alpha_f^{a_2\to n} \otimes \alpha_f^{a_3\to n} \otimes \dots \tag{4.2}$$

for any subset of the diamond ancestors $a_1, a_2, a_3, \dots$ of n in $\mathcal{G}(f)$ is also an arrival curve for f at the output of the PEF, where $\otimes$ denotes the min-plus convolution (Definition 2.2).

Item 2/ is proven by considering the entire system between the diamond ancestor a and the vertex n. This system is causal, but is neither lossless nor FIFO. As we focus on obtaining an output arrival curve, several classic network-calculus results remain valid, which we detail in the formal proof in Appendix B.2.2. Several references have also studied non-lossless systems, but when focusing on the service curves [Ciucu, Schmitt, Wang 2011].

Example: An arrival curve $\alpha_{f,\mathrm{PEF}^*}$ for f at the output of the PEF within $\mathtt{S3}_{\mathtt{South}}$ (Figure 4.3) is shown in Figure 4.12 with a solid red line.

The first constituent, $\alpha_{f,\mathrm{PEF}^{\mathrm{in}}}$ is the arrival curve at f at the input of the PEF (as per Theorem 4.1, Item 1/). To obtain it, we first take the arrival curve $\alpha_{f,\mathtt{S5}^*_{\mathtt{North}}} = \gamma_{r_0,b_0}$ of f at the output of $\mathtt{S5}_{\mathtt{North}}$. We then use the jitter bound within each path in Figure 4.3 and apply Lemma B.2 in Appendix B.2.1 to obtain the arrival curves $\alpha_{f,\mathtt{S2}^*_{\mathtt{South}}}$ and $\alpha_{f,\mathtt{S4}^*_{\mathtt{East}}}$: they both equal $\gamma_{r_0,2b_0}$. As f enters $\mathtt{S3}_{\mathtt{South}}$ from both $\mathtt{S2}_{\mathtt{South}}$ and $\mathtt{S4}_{\mathtt{East}}$, we obtain $\alpha_{f,\mathrm{PEF}^{\mathrm{in}}} = \alpha_{f,\mathtt{S2}^*_{\mathtt{South}}} + \alpha_{f,\mathtt{S4}^*_{\mathtt{East}}} = \gamma_{2r_0,4b_0}$.

The second constituent of $\alpha_{f,\mathrm{PEF}^*}$ in Figure 4.12 is obtained by applying the Equation (4.1) of Theorem 4.1, Item 2/ with $a = \mathtt{S5}_{\mathtt{North}}$. From Figure 4.3, we obtain that a delay lower-bound [resp., an upper-bound] for f from $\mathtt{S5}^*_{\mathtt{North}}$ to $\mathtt{S3}^{\mathrm{in}}_{\mathtt{South}}$ along any possible paths within $\mathcal{G}(f)$ is $d_f^{\mathtt{S5}^*_{\mathtt{North}}\to\mathtt{S3}^{\mathrm{in}}_{\mathtt{South}}} = 0$ t.u. [resp., $D_f^{\mathtt{S5}^*_{\mathtt{North}}\to\mathtt{S3}^{\mathrm{in}}_{\mathtt{South}}} = 7$ t.u.]. We obtain $\alpha_f^{\mathtt{S5}_{\mathtt{North}}\to\mathtt{S3}_{\mathtt{South}}} = \alpha_{f,\mathtt{S5}^*_{\mathtt{North}}} \oslash \delta_{D_f^{\mathtt{S5}^*_{\mathtt{North}}\to\mathtt{S3}^{\mathrm{in}}_{\mathtt{South}}} - d_f^{\mathtt{S5}^*_{\mathtt{North}}\to\mathtt{S3}^{\mathrm{in}}_{\mathtt{South}}}} = \gamma_{r_0,b_0} \oslash \delta_7$, *i.e.*, $\alpha_f^{\mathtt{S5}_{\mathtt{North}}\to\mathtt{S3}_{\mathtt{South}}} = \gamma_{r_0,8b_0}$.

If we assume that the PEF does not delete any packet, as in the *intuitive* approach mentioned in Section 4.1.3, we only know that f has the arrival curve $\alpha_{f,\mathrm{PEF}^{\mathrm{in}}}$ at the output of the PEF (Item 1/ of the Theorem).

But our theorem goes beyond the intuitive approach: its second item applied with $a = \mathtt{S5}_{\mathtt{North}}$ provides a second arrival curve for f: $\alpha_f^{\mathtt{S5}_{\mathtt{North}}\to\mathtt{S3}_{\mathtt{South}}}$. We apply Theorem 2.1 to combine the knowledge of the two arrival curves: $\alpha_{f,\mathrm{PEF}^*} = \alpha_{f,\mathrm{PEF}^{\mathrm{in}}} \otimes \alpha_f^{\mathtt{S5}_{\mathtt{North}}\to\mathtt{S3}_{\mathtt{South}}}$ is also an arrival curve for f at the output of the PEF. Theorem 4.1 provides a better upper-bound of the traffic than the intuitive approach. For example, $\alpha_{f,\mathrm{PEF}^*}$ indicates that the double rate $2r_0$ is only a peak rate that the traffic cannot exhibits forever: flow f keeps a sustained rate r_0, but with a much higher burst $8b_0$.

Theorem 4.1 does not require to identify pairs of replication/elimination functions, with one PRF and one PEF in each pair. Therefore, the result is suited for complex flow graphs,

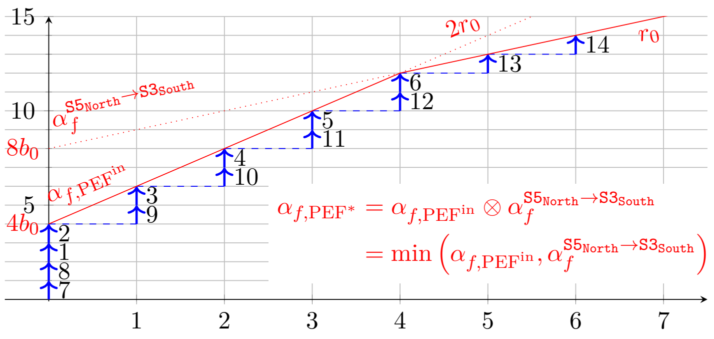


Figure 4.12: Solid red: Arrival curve of f on the toy example, at the output of $\mathtt{PEF}_{\mathtt{S3_{South}}}(f)$, obtained by using Theorem 4.1. Dashed blue: Cumulative arrival function obtained with the trajectory of Fig. 4.14, showing the tightness of the result.

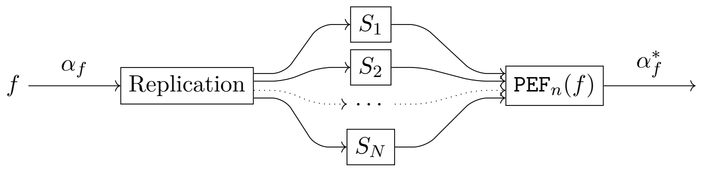


Figure 4.13: Notations of Corollary 4.1. Flow f is replicated and sent to N parallel systems. Corollary 4.1 gives the arrival curve α_f^* at the output of the packet-elimination function $\mathtt{PEF}_n(f)$.

including graphs with repeated patterns of redundancy, with meshes, as well as graphs where the packet-elimination function is not located at the merge point of the paths. Chapter 6 describes how the identification of diamond ancestors is performed in the xTFA tool.

When pairs of PRF/PEF can be identified as in Figure 4.2, we can use:

Corollary 4.1 (Unique redundant section with parallel systems)
Consider a flow f with an arrival curve α_f that is replicated and sent into N causal systems $\{S_i\}_{i\in[\![1,N]\!]}$ and then processed by a packet-elimination function $\mathtt{PEF}_n(f)$, as in Figure 4.13. Note that each S_i is not necessary a single network element but can be any combination of network elements. Assume that the packets forwarded through S_i (i.e., *the ones not lost) have a delay through S_i that is bounded within $[d_i, D_i]$. Then,*

$$\alpha_f^* = \left(\sum_{i\in[\![1,N]\!]} \alpha_f \oslash \delta_{(D_i-d_i)}\right) \otimes \left(\alpha_f \oslash \delta_{\left(\max\limits_{i\in[\![1,N]\!]} D_i - \min\limits_{j\in[\![1,N]\!]} d_j\right)}\right) \tag{4.3}$$

is an arrival curve for f at the output of $\mathtt{PEF}_n(f)$.

The Corollary is a direct application of Theorem 4.1, with the corresponding notations. Its formal proof is in Appendix B.2.3. Corollary 4.1 is of interest for two reasons. First, its simpler notation is likely to cover many industrial applications containing a unique redundant portion with parallel systems. Second, Corollary 4.1 is tight in the following sense.

Proposition 4.1 (The result in Corollary 4.1 is tight with $N = 2$ and leaky-bucket-constrained flows, in the family of VBR arrival curves)
For any *leaky-bucket arrival curve $\gamma_{r,b}$, for any set of values $d_1, D_1, d_2, D_2 \in \mathbb{R}$ such*

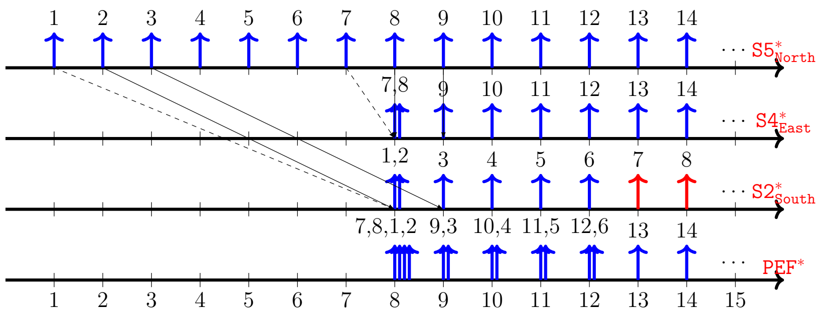


Figure 4.14: Trajectory showing that the result of Corollary 4.1 is tight for the toy example. The cumulative function of f, starting at Time Unit 8 in the above trajectory, is given as a dashed blue line in Figure 4.12.

that $d_1 \leq D_1$ and $d_2 \leq D_2$,
***there exists** a flow f with arrival-curve $\alpha_f = \gamma_{r,b}$ and no minimum packet length whose content is replicated and sent to two systems S_1 and S_2 in which the packets of f suffer a delay bounded in $[d_1, D_1]$ and $[d_2, D_2]$ respectively; the sum of the outputs of the two systems is then processed by a packet-elimination function PEF$_n(f)$,*
***such that**, the arrival curve α^*_f defined in (4.3) is the best VBR arrival curve (Figure 2.1c) for f at the output of PEF$_n(f)$.*

The formal proof is in Appendix B.2.4. We give here an intuition on the toy example.

Example: Our goal is to obtain a cumulative function $R_{f,\texttt{PEF}^*}(t)$ at the output of PEF such that $t \mapsto R_{f,\texttt{PEF}^*}(t) - R_{f,\texttt{PEF}^*}(s)$ 'perfectly fits' the arrival curve $\gamma_{2r_0,4b_0} \otimes \gamma_{r_0,8b_0}$, for some observation starting time s (as in Figure 4.12).

This is done by using the trajectory shown in Figure 4.14. In the figure we spread the packets within t.u. 8 for ease of reading, but they exit at the exact same time. Because of this, we can also put an arbitrary order of arrivals among them.

If we start counting the packets at t.u. 8, we observe the cumulative arrival function shown in dashed blue in Figure 4.12, for which it is clear that the arrival curve in solid red is the best VBR envelope.

4.3.2 Reordering Introduced by the Packet Replication and Elimination

In Section 4.3.1 we provide a characterization of the traffic at the output of a PEF in the form of an arrival curve. The arrival curve can then be used to compute delay and backlog bounds on subsequent vertices, from which we can obtain the ETE delay bounds. However, the data units at the output of the PEF are also out-of-order. The mis-ordering of the flow's data units cannot be captured by the notion of arrival curves. But it still has an effect on the performances of time-sensitive networks: some applications require in-order-delivery or a bounded out-of-order delivery.

Two metrics are of interest for quantifying the mis-ordering in time-sensitive networks.

Definition 4.5 (Rordering late Time Offset, Reordering Byte Offset. [Mohammadpour, Le Boudec 2021], [RFC 4737]) *Consider a flow f and two vertices n, o of $\mathcal{G}(f)$ such that o is a diamond ancestor of n in $\mathcal{G}(f)$ and n is not an EP-vertex of $\mathcal{G}(f)$. Consider an observation point v [resp., w] of n [resp., n] such that f is packetized at v and w. Define*

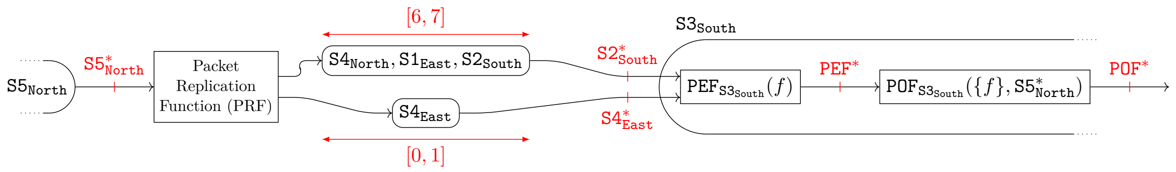


Figure 4.15: Toy example of Figure 4.3, with a packet-ordering function (POF) placed after the PEF to correct the mis-ordering caused by the redundancy.

m_k as the data unit of f that crosses w in the k-th position and denote by l_k its size and by E_k its arrival time at v (with the convention $E_k = +\infty$ if m_k is lost for v).
Then the reordering late time offset (RTO) $\lambda_{f,v}(w)$ and the reordering byte offset (RBO) $\pi_{f,v}(w)$ of f at v with respect to w are

$$\lambda_{f,v}(w) \triangleq \sup_{k\geq 0}\left(E_k - \min_{j|j>k} E_j\right) \qquad \pi_{f,v}(w) \triangleq \sup_{k\geq 0}\left(\sum_{j|j>k, E_j<E_k} l_j\right) \tag{4.4}$$

If $\texttt{POF}_n(\{f\}, o^*)$ is a packet-ordering function that forces the data units of f to be in the same order as their order at the output of o, then, $\lambda_{f,\texttt{POF}_n^{\text{in}}}(o^*)$ gives the minimum timeout value of the POF algorithm and $\pi_{f,\texttt{POF}_n^{\text{in}}}(o^*)$ gives its required buffer size [Mohammadpour, Le Boudec 2021]. In general, if a destination d does not support any out-of-order delivery, then a function $\texttt{POF}_d(\{f\}, \phi)$ is placed just before delivery to the application. The end-to-end RTO $\lambda_{f,d^{\text{in}}}(\phi)$ and RBO $\pi_{f,d^{\text{in}}}(\phi)$ must be obtained to correctly configure this POF.

Proposition 4.2 (RBO $\leq \alpha$(RTO))
For a flow f, and two observations points v, w as in Definition 4.5, if $\lambda_{f,v}(w) < +\infty$,

$$\pi_{f,v}(w) \leq \alpha_{v,f}(\lambda_{f,v}(w)) \tag{4.5}$$

The proof in Appendix B.2.5 follows directly from Definitions 2.5 and 4.5. Proposition 4.2 combined with Theorem 4.1 show that we can focus on the effect of the PEF on the RTO to also obtain a bound on the RBO.

Theorem 4.2 (RTO at the output of a PEF)
Consider a flow f, a vertex n containing a packet-elimination function $\mathit{PEF}_n(f)$ and a diamond ancestor a of n in $\mathcal{G}(f)$. Denote by $d_f^{a^*\to n^{in}}$ [resp., $D_f^{a^*\to n^{in}}$] a lower [resp., upper] delay bound for f between the output of a and the input of $\mathit{PEF}_n(f)$, along any possible path in the graph $\mathcal{G}(f)$. Then $\lambda_{f,\mathit{PEF}_n(f)^*}(a^*)$ verifies

$$\lambda_{f,\mathit{PEF}_n(f)^*}(a^*) \leq \left| D_f^{a^*\to n^{in}} - d_f^{a^*\to n^{in}} - \alpha^{\downarrow}_{f,a*}(2L^{min})\right|^+ \tag{4.6}$$

where $|x|^+ \triangleq \max(0, x)$, $L^{\min}$ is the minimum packet size of f, α_{f,a^*} is an arrival curve for f at the output of a and $\alpha^{\downarrow}_{f,a^*}$ is its lower pseudo-inverse defined by $\alpha^{\downarrow}_{f,a^*}(y) = \inf\{x|\alpha_{f,a^*}(x) \geq y\}$ [Liebeherr 2017, §10].

The proof in Appendix B.2.6 relies on [Mohammadpour, Le Boudec 2021, Thm. 5].

Example: On the toy example of Figure 4.3, $\alpha^{\downarrow}_{f,\texttt{S5}^*_{\texttt{North}}}(2L_{\min}) = 1$ t.u. Theorem 4.2 proves that the RTO at the output of the PEF in Figure 4.3 is bounded by 6 t.u. In Figure 4.14,

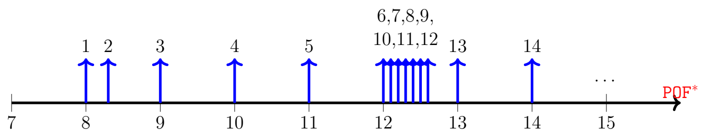


Figure 4.16: Trajectory of the packets at the output of the POF of Figure 4.15 when the POF processes the packets from the trajectory of Figure 4.14.

we observe that Data Unit 6 is late by 4 time units with respect to Data Unit 7. The RTO of f is hence between 4 and 6 time units.

Assume now that we place, after the PEF, the function $\texttt{POF}_{\texttt{S3}_{\texttt{South}}}(\{f\}, \texttt{S5}^*_{\texttt{North}})$, a packet-ordering function enforcing for f the order defined at $\texttt{S5}^*_{\texttt{North}}$ (Figure 4.15). The RTO bound that we computed above gives the minimum timeout value $T^{\texttt{POF}}$ (6 t.u.) and by combining with Proposition 4.2, we obtain a bound on the RBO (14 d.u.) which gives the minimum buffer size of the POF. In the trajectory of Figure 4.14, the POF receives the traffic from the Line $\texttt{PEF}^*$ and forces the data units to be in the same order as on the Line $\texttt{S5}^*_{\texttt{North}}$. The resulting output is given in Figure 4.16.

We note that all data units continue to have a delay upper-bounded by 7 time units. Indeed, when no data units is lost for the POF, then the latter does not increase the end-to-end delay of the data units [Mohammadpour, Le Boudec 2021, Thm. 4].

Second, we observe that the traffic at the output of the POF (Figure 4.16) is much more bursty than the traffic at the output of the PEF (Line $\texttt{PEF}^*$ in Figure 4.14) and is no longer constrained by $\gamma_{2r_0,4b_0} \otimes \gamma_{r_0,8b_0}$.

Corollary 4.2 (Arrival curve at the output of a POF, [Mohammadpour, Le Boudec 2021, Corollary 1])
Consider f a flow and a, n two vertices of $\mathcal{G}(f)$ such that n is not an EP-vertex of $\mathcal{G}(f)$ and a is a diamond ancestor of n in $\mathcal{G}(f)$. Assume that $\mathit{POF}_n(\{f\}, a^*)$ is a POF for f at n, and denote by $d_f^{a^*\to n^{in}}$ [resp., $D_f^{a^*\to n^{in}}$] a lower [resp., upper] delay bound for f between the output of a and the input of the POF in n. If the sequence of data unit that reaches the POF is not incomplete (no data unit is lost for the POF), then

$$\alpha_{f,\mathit{POF}_n(\{f\},a)^*} = \alpha_{f,a^*} \oslash \delta_{D_f^{a^*\to n^{in}} - d_f^{a^*\to n^{in}}}$$

is an arrival curve for f at the output of the POF. If the sequence can be incomplete, then

$$\alpha_{f,\mathit{POF}_n(\{f\},a)^*} = \alpha_{f,a^*} \oslash \delta_{D_f^{a^*\to n^{in}} - d_f^{a^*\to n^{in}} + T^{\mathit{POF}}}$$

is an arrival curve for f at the output of the POF, with T^{POF} the value of the timeout of the POF.

Corollary 4.2 is a direct application of [Mohammadpour, Le Boudec 2021, Cor.1]. Placing a POF after a PEF hence comes with benefits and drawbacks, as summarized on the first line of Table 4.3.

4.3.3 Obtaining End-to-End Delay Bounds in Networks with PREFs

If a vertex n of the graph induced by flows (GIF) contains a PEF or/and a POF, then we can obtain an arrival curve $\alpha_{n^\dagger}$ for the class of interest at the input of the CBQS by applying the results of Sections 4.3.1 and 4.3.2. From $\alpha_{n^\dagger}$ and from the service curve β_n of the CBQS within n (assumed known, see Section 2.6.4), we can derive the delay bounds $[d_n, D_n]$ through n as well as the individual arrival curves of each flow at n' and n^*, using the same techniques as in Chapter 3, Section 3.3.

But computing $\alpha_{n^\dagger}$ does not only require the knowledge of the individual arrival curves: For a flow f for which a PEF is installed at n, for each diamond ancestor a, applying Item 2/ of Theorem 4.1 requires the knowledge of the arrival curve α_{f,a^*}, as well as an upper delay bound [resp., a lower delay bound] for f from a to n along any possible path. Chapter 6 details how to store and obtain this information for feed-forward networks.

But for networks with cyclic dependencies, we can simplify the above issue and choose to apply, for each flow f processed by a PEF, the Item 2/ of Theorem 4.3.1 using only the source ϕ of f as a diamond ancestor. Computing the effect of the PEF with this single diamond ancestor requires only the knowledge of the burst $b_{f,p'}$ of the flow f at the output of each parent p, as well as the bounds $[d_f^{\phi\to p^*}, D_f^{\phi\to p^*}]$ for the delay of f between the source of f and p^*. Overall, computing the node n in the TFA approach is an application

$$\mathcal{F}_n : \boldsymbol{b_n}, \boldsymbol{d_n} \mapsto \boldsymbol{b'_n}, \boldsymbol{d'_n} \tag{4.7}$$

where

$$\boldsymbol{b_n} = (b_{f,p'})_{\forall p \text{ parent of } n; \forall f | f \ni (p,n)}, (b_{f,\phi})_{\forall f | \text{source}(f)=n}$$

is the vector of the burstiness within each parent p of each flow f that enters n from p, plus the initial burstiness of all flows generated within n, and

$$\boldsymbol{d_n} = (D_f^{\phi\to p^*})_{\forall p \text{ parent of } n; \forall f | f \ni (qpn)}, (0)_{\forall f | \text{source}(f)=n}$$

is the vector of delay upper-bounds between the source ϕ of f and the output p^* of the parent p, along any possible path in $\mathcal{G}(f)$. This vector is extended with a vector full of zeros for all the flows whose source is the current vertex n (they have not suffered any jitter so far).

For a network with cyclic dependencies, we first perform a static analysis of the network (not based on network calculus) to obtain the lower-bounds on the minimum delays $d_f^{\phi\to v}$ for any flow f and any flow v. This analysis is based, for example, on the minimum propagation time, the lower-bounds on the technological latencies, etc. Then we perform the cuts as in Chapter 3. The combination of the $\mathcal{F}_n$ applications for the virtual feed-forward network provides and algorithm $\mathcal{FF}$

$$\mathcal{FF}(\boldsymbol{b}, \boldsymbol{d}) \mapsto (\boldsymbol{b'}, \boldsymbol{d'})$$

that maps the burst and delay bounds after the cuts to the burst and delay bounds before the cuts. As in Chapter 3, we have the following result.

Theorem 4.3 (Extended fixed-point result: Validity of any non-negative fixed-point $(\overline{\boldsymbol{b}}, \overline{\boldsymbol{d}})$)
Any non-negative fixed-point $(\overline{\boldsymbol{b}}, \overline{\boldsymbol{d}})$ *of* $\mathcal{FF}$*, ie such that* $\mathcal{FF}(\overline{\boldsymbol{b}}, \overline{\boldsymbol{d}}) = (\overline{\boldsymbol{b}}, \overline{\boldsymbol{d}})$*, constitute*

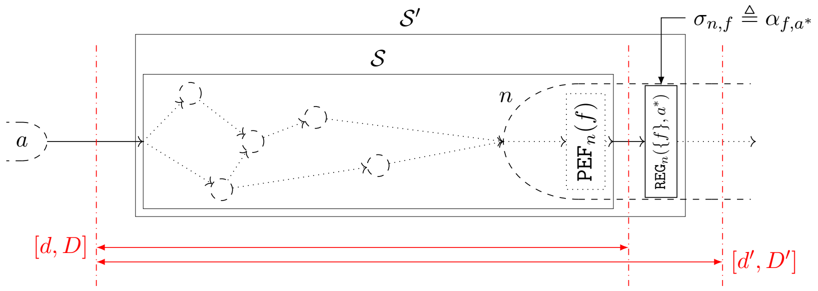


Figure 4.17: Notations for the analysis of the interactions between PEF and a PFR for a flow f. Vertices of $\mathcal{G}(f)$ are shown in dashed circles/ovals and edges are shown with dotted arrows.

a valid bound for the bursts and the jitter bounds from source, at the cuts for the real network with cyclic dependencies. If $\mathcal{FF}$ has a non-negative fixed-point, then the network is stable.

The formal proof in Appendix B.2.7 is very similar to the one for Theorem 3.2. Chapter 6 details how to apply Theorem 4.3.

4.4 Interactions between PREF and Traffic Regulators

Subsection 4.3.2 shows that a packet-ordering function (POF) can be used after a PEF to remove the mis-ordering caused by the redundancy. Similarly, regulators (REGs) can be used after a PEF to remove the burstiness increase caused by the redundancy. But regulators are queuing systems and their effect on the worst-case ETE delay should be accounted for.

In this section, we first analyze the interactions between PEF and a regulator placed directly after. We evaluate how these interactions affect the ETE delay guarantees of the flows, and we show that the conclusions highly depend on the nature of the regulator (either PFR or IR). We last analyze the effect of a POF placed after the PEF and before the REG.

4.4.1 Delay-Bound Analysis of PREFs Combined with Per-Flow Regulators

Consider a vertex n containing a function $\texttt{PEF}_n(f)$ and consider a diamond ancestor a of n in $\mathcal{G}(f)$ (Figure 4.17). Consider the system $\mathcal{S}$ between the output of a and the output of $\texttt{PEF}_n(f)$ (solid box in Figure 4.17). Due to all the possible paths with different lengths, $\mathcal{S}$ is neither FIFO nor lossless. However, $\mathcal{S}$ is causal because a is a diamond ancestor of n in $\mathcal{G}(f)$.

We denote by d [resp., D] a delay lower-bound [resp., upper-bound] for each forwarded data unit through $\mathcal{S}$. The PEF has no delay, hence $d = d_f^{a^*\rightarrow n^{\text{in}}}$ and $D = D_f^{a^*\rightarrow n^{\text{in}}}$.

After $\mathcal{S}$, and still within vertex n (dashed oval on the right of Figure 4.17), we place a per-flow regulator: $\texttt{REG}_n(\{f\}, a^*)$ with shaping curve $\sigma_{n,f} \triangleq \alpha_{f,a^*}$. We now consider the system $\mathcal{S}'$ made of $\mathcal{S}$ followed by the PFR, and we are interested in the delay bounds $[d', D']$ for the non-lost data units through $\mathcal{S}'$. If $\mathcal{S}$ were FIFO, we could use the essential *shaping-for-free* introduced in Chapter 3, Section 3.1.5, and we would have $D' = D$. But, as $\mathcal{S}$ is not FIFO, the PFR does not guarantee the *shaping-for-free* property, as we show on the toy example.

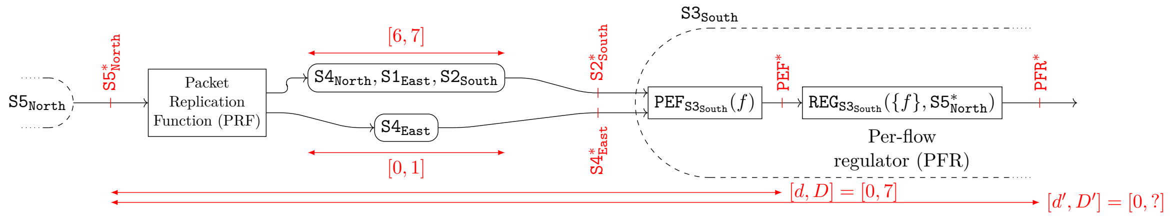


Figure 4.18: Toy example of Figure 4.3 with a per-flow regulator (PFR) placed after the PEF to remove the burstiness increase caused by the redundancy.

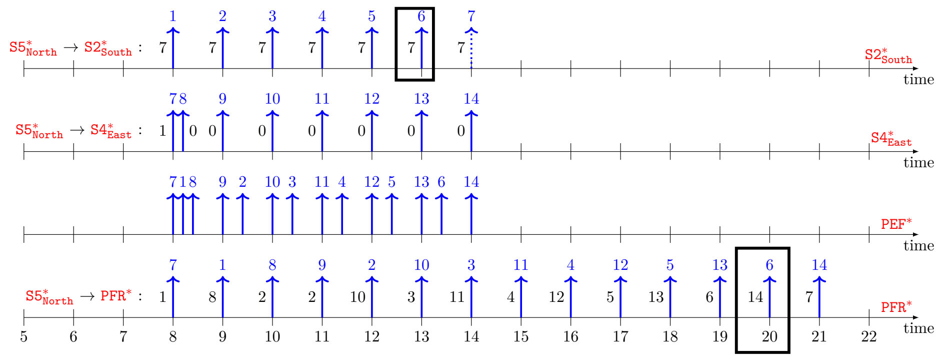


Figure 4.19: An acceptable trajectory on the toy example, that shows that the delay bound D' through $\mathcal{S}'$ is at least 14 t.u. The delay of the data units from $\mathtt{S5}^*_{\mathtt{North}}$ to the observation points are given on the left of the packets.

Example: Figure 4.18 considers the toy example from Figure 4.3, to which we add $\mathtt{REG}_{\mathtt{S3_{South}}}(\{f\}, \mathtt{S5}^*_{\mathtt{North}})$ within vertex $\mathtt{S3_{South}}$, just after the function $\mathtt{PEF}_{\mathtt{S3_{South}}}(f)$. System $\mathcal{S}$ is between the observation points $\mathtt{S5}^*_{\mathtt{North}}$ and $\mathtt{PEF}^*$ and $[d, D] = [0, 7]$ t.u. $\mathcal{S}'$ is between the observation points $\mathtt{S5}^*_{\mathtt{North}}$ and $\mathtt{PFR}^*$.

Figure 4.19 presents an acceptable trajectory. The traffic profile at $\mathtt{S5}^*_{\mathtt{North}}$, not shown, is periodic as in Figure 4.14. The line $\mathtt{PEF}^*$ gives the resulting trajectory at the output of the PEF.

Based on its input $\mathtt{PEF}^*$ and on its shaping curve ($\sigma_{\mathtt{S3_{South}},f} = \alpha_{f,\mathtt{S5}^*_{\mathtt{North}}} = \gamma_{r_0,b_0}$), the PFR outputs the packets as shown on the Line $\mathtt{PFR}^*$.

We observe that the d.u. 6 suffers through $\mathcal{S}'$ a total delay of 14 t.u., *i.e.*, twice the delay bound D through $\mathcal{S}$ alone. We note that this can be explained by the time needed by the PFR for processing d.u. 1 to 5 and 7 to 13. This is done even though d.u. 7 to 13 are out of order ("too early") with respect to d.u. 6. At $\mathtt{PFR}^*$, the packet containing d.u. 6 is late with respect to d.u. 7 by 12 t.u. units, thus the RTO of f is at least 12 t.u.

We observe that the output of the PEF is bursty and out of order, and the PFR placed afterwards paces the packets to remove the burstiness. But, by doing so, the PFR worsens the mis-ordering of the packets (12 instead of 6) and increases the delay of the late packets (Packet 6), thus increasing the worst-case ETE delay (at least 14 time units). As such, the regulator comes with a *delay penalty*, that we can upper-bound:

Theorem 4.4 (Bound on the delay penalty of a PFR placed after a PEF)
With the notations of the subsection, assume that the PFR $\mathtt{REG}_n(\{f\}, a^*)$ *is configured*

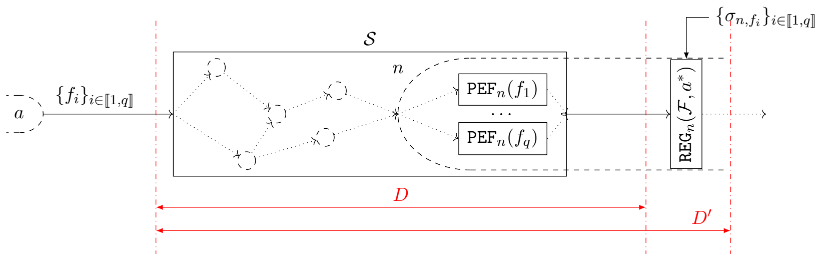

$\{\sigma_{n,f_i}\}_{i\in[\![1,q]\!]}$
$\mathcal{S}$
n
a
$\{f_i\}_{i\in[\![1,q]\!]}$
$\texttt{PEF}_n(f_1)$
$\cdots$
$\texttt{PEF}_n(f_q)$
$\texttt{REG}_n(\mathcal{F},a^*)$
D
D'

Figure 4.20: Notations for the analysis of the interactions between PEFs and an interleaved regulator (IR) for an aggregate of flows $\mathcal{F} = \{f_i\}_{i\in[\![1,q]\!]}$.

with a leaky-bucket shaping curve $\sigma_{n,f} = \gamma_{r,b}$ (with b greater than the maximum packet size of f) and that $\sigma_{n,f}$ is an arrival curve of f at the input of $\mathcal{S}$. If D [resp., d] is an upper [resp., a lower] bound on the delay of f through the system $\mathcal{S}$ (Figure 4.17), then $D' = 2D - d$ [resp., $d' = d$] is an upper [resp., a lower] bound on the delay of f through the system $\mathcal{S}'$.

The formal proof in Appendix B.2.8 relies on the service-curve characterization of a PFR (Proposition 3.1). The combination of Theorem 4.4 with [Mohammadpour, Le Boudec 2021, Thm. 7] gives directly the following result.

Corollary 4.3 (Bound on the RTO at the output of a PFR placed after a PEF)
With the notations of Theorem 4.4, the RTO $\lambda_{n,\textit{PFR}^*}(f,a^*)$ of f at the output of $\textit{PFR}_n(\{f\},a^*)$, with reference a^*, verifies

$$\lambda_{n,\textit{PFR}^*}(f,a^*) \leq \lambda_{n,\textit{PEF}^*}(f,a^*) + D - d$$

with $\lambda_{n,\textit{PEF}^*}(f,a^*)$ the RTO of f at the output of the PEF, again with respect to the order of the data units at a^*.

Example: Applying Theorem 4.4 shows that $2D - d = 14$ t.u. is an upper delay bound through $\mathcal{S}'$. As it is achieved by Data Unit 6 in Figure 4.19, it is also the worst-case delay. Applying Corollary 4.3 to the toy example gives that 13 t.u. is an upper-bound on the RTO of the flow at the output of the PFR, with respect to the order of the packets at $\texttt{S5}^*_{\texttt{North}}$. Data Unit 6 in the trajectory achieves a reordering offset of 12 time units (with respect to Data Unit 7), Thus the worst-case RTO at the output of $\mathcal{S}'$ in the toy example is between 12 and 13 time units.

When a PFR is used after a PEF, the current subsection shows that the *shaping-for-free* property does not hold, but Theorem 4.4 captures the delay penalty by using the service-curve characterization of PFRs (Proposition 3.1), combined with the arrival curve obtained from Theorem 4.1. As we do not know any service-curve characterization for an IR, Theorem 4.4 cannot apply to IRs.

4.4.2 Instability of the Interleaved Regulator Placed after a Set of PEFs

With an interleaved regulator (IR), several flows $\mathcal{F} = \{f_i\}_{1\leq i\leq q}$ that share the same redundant section $a \to n$ are processed by the same IR $\texttt{REG}_n(\mathcal{F},a^*)$, after their respective elimination function $\texttt{PEF}_n(f_i)$ for $i \in [\![1,m]\!]$ (see Figure 4.20).

When the aggregate $\mathcal{F}$ contains a unique flow, then the IR is a PFR. Therefore, we do not expect the *shaping-for-free* property to be valid with the IR either. But when the IR processes several flows, we exhibits an adversarial model that yields unbounded latencies:

Theorem 4.5 (Instability of the IR placed after the PEFs)
Consider a network with graph $\mathcal{G}$ and consider $q \in \mathbb{N}$ flows $f_1, \dots, f_q$ (see Figure 4.20). Take two vertices a and n such that, for each $i \in [\![1, q]\!]$, a is a diamond ancestor of n in $\mathcal{G}(f_i)$. Assume that

(a) for each $i \in [\![1, q]\!]$, vertex n contains $\texttt{PEF}_n(f_i)$, a PEF for f_i,

(b) vertex n contains $\texttt{REG}_n(\{f_i\}_{i \in [\![1,q]\!]}, a^*)$, an interleaved regulator for the aggregate, placed after the PEFs, with the same leaky-bucket shaping curve for each flow: $\forall i \in [\![1, q]\!], \sigma_{f_i,n} = \gamma_{r,b}$,

(c) all graphs $\{\mathcal{G}(f_i)\}_{i \in [\![1,q]\!]}$ share at least two different paths P_1, P_2 to reach n from a.

Then, for $q \in \mathbb{N}$ and $r, b, d_1, d_2, D_1, D_2 \in \mathbb{R}^+$ with $d_1 \leq D_1$, $d_2 \leq D_2$ and $D_1 \leq D_2$ (flipping the indexes if required), if

(d) b is greater than the minimum packet length,

(e) d_1, D_1, d_2, D_2 are not all equal and $q \geq q_{\min}$ with

$$q_{\min} \triangleq \left\lfloor \frac{2r\,|d_2 - D_1|^+}{b} + 2 \right\rfloor + 1 \tag{4.8}$$

then there exists an adversarial traffic arrival at a^* for each of the q flows and an adversarial implementation of the paths $\{P_j\}_j$ such that

1/ each flow f_i is $\gamma_{r,b}$-constrained at a^*,

2/ for each data unit m belonging to one of the flows $\{f_i\}_{i \in [\![1,q]\!]}$, if m is not lost on P_1 [resp., on P_2], then its delay along P_1 [resp., along P_2] is within $[d_1, D_1]$ [resp., within $[d_2, D_2]$],

3/ flows $\{f_i\}_i$ have an unbounded latency within the IR,

4/ P_1 and P_2 are both FIFO,

5/ the system $\mathcal{S}$ made of the sub-graph of $\mathcal{G}$ between a^* and the output of the PEFs (Figure 4.20) remains lossless and FIFO-per-flow for each f_i.

The proof is in Appendix B.2.9. In a real-life system, the backlog of the IR cannot increase indefinitely: Its buffer will overflow at some time, causing congestion losses thus breaking a major requirement of time-sensitive networks.

Note that only Properties 1/ to 3/ of Theorem 4.5 are required to prove the validity of the adversarial model. However, our adversarial model provides additional Properties 4/ and 5/; they are of interest when considering the solutions for preventing the instability, as we illustrate in Sec. 4.4.3. Theorem 4.5 also provides a mean to obtain the following wider result, whose proof is in Appendix B.2.10.

Table 4.3: Benefits and Drawbacks of Several Configurations, Compared to the Situation with the PEF(s) only.

Config	Benefits with respect to the PEF alone	Drawbacks with respect to the PEF alone
PEF + POF	• Destination receives the data units in order • Reordering-for-free: the POF does not increase the end-to-end delay bounds.	• The POF worsens the arrival curve; this can lead to higher delay bounds in downstream nodes. • Increased hardware complexity (Figure 4.8).
PEF + REG	• Output traffic keeps the arrival constraints it had before the redundant section, resulting in smaller delay bounds in downstream nodes.	• Delay penalty due to mis-ordering: with PFR: delay penalty with a guaranteed maximum delay; with IR: unbounded delay. • Increased hardware complexity (Figure 4.9).
PEF + POF + REG	• Destination receives the data units in order • Reordering-for-free and shaping-for-free: [POF + REG] does not increase the delay bounds. • Output traffic keeps the same arrival constraints as it had before the redundant section.	• Increased hardware complexity (Figures 4.8 and 4.9).

Corollary 4.4 (Instability of the interleaved regulator after a non-FIFO system, even if the system is FIFO-per-flow and lossless)
For any $D_{\max} > 0$, $r > 0$, b greater than the minimum packet length, and for any IR that processes three or more flows $\{f_i\}_i$ using the same leaky-bucket shaping curve $\gamma_{r,b}$, there exists a lossless FIFO-per-flow system $\mathcal{S}$ and a $\gamma_{r,b}$-constrained adversarial generation of each flow at the input of $\mathcal{S}$ such that, when the IR is placed after $\mathcal{S}$, the delay of the flows through $\mathcal{S}$ is upper-bounded by $D_{\max}$ but the delay of the flows through the IR is not bounded.

4.4.3 Effect of the Packet-Ordering Function on the Combination of a PEF with Traffic Regulators

Table 4.3 summarizes the benefits and drawbacks of using regulators after a PEF, as analyzed in Sections 4.4.1 and 4.4.2. We observe that the drawbacks of the regulators appear symmetrical with respect to those of the POF. For example, a main issue of the POF is the burstiness of the traffic at its output; this can be corrected by using a regulator. A main issue of the REGs is the delay penalty caused by the out-of-order input; this can be solved by placing a POF just before.

The combination PEF + POF + REG appears as a potential solution for keeping the benefits of both the POF and the REG without their main drawbacks. We first analyze this new configuration on the toy example.

Example: We go back to the toy example of Figure 4.18, *i.e.*, with a unique flow f and a periodic profile of one data unit every time unit. We add a POF before the PFR. It gives the situation presented in Figure 4.21. The POF is configured to enforce the order of the data units as seen at $\texttt{S5}^*_{\texttt{North}}$. Assume for example that it receives the traffic defined by the line PEF* of Figure 4.19. Then the POF outputs the data units as on the Line POF* of Figure 4.22. The PFR further processes this trajectory to spread the data units as per the flow's contract and outputs them as on the Line PFR* of Figure 4.22. The resulting traffic is constrained with the initial arrival curve α_{r_0,b_0}. We observe that all the data units have kept an ETE delay below 7 t.u.

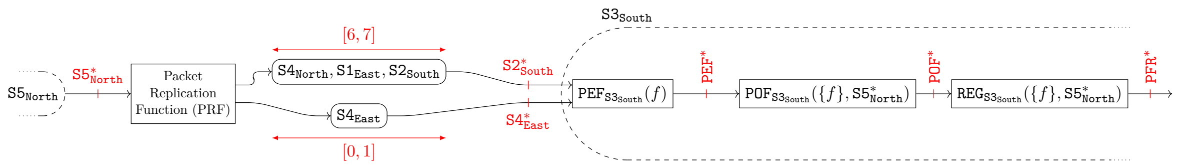


Figure 4.21: Toy example of Figure 4.3, to which we added a POF followed by a PFR.

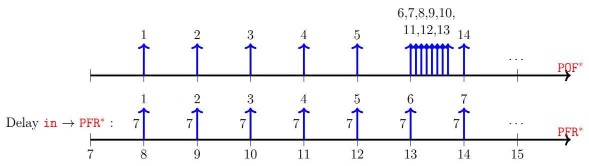


Figure 4.22: Output of the POF and of the PFR of Figure 4.21 when they process the trajectory of Figure 4.19.

When using an interleaved regulator, Property 5/ of Theorem 4.5 shows that the re-sequencing must be performed globally on the aggregate processed by the IR, and not for each flow individually. Indeed, Property 5/ of Theorem 4.5 shows that per-flow reordering has no effect on the trajectory generated by the adversarial model, and the instability remains.

When re-sequencing the aggregate globally, we obtain the following result, valid for both PFRs and IRs.

Theorem 4.6 (Elimination-resequencing-reshaping is for free)
Consider a network with graph $\mathcal{G}$ and consider a set of one or more flows $\mathcal{F}$. Take a and n two vertices of $\mathcal{G}$ such that for each flow $f \in \mathcal{F}$, a is a diamond ancestor of n in $\mathcal{G}(f)$ (see Figure 4.23). Assume that the CBQS within n is preceded by the following functions, in this order: a set of parallel packet-elimination functions $\{\mathtt{PEF}_n(f)\}_{f\in\mathcal{F}}$, followed by a unique packet-ordering function with configuration $\mathtt{POF}_n(\mathcal{F}, a)$, and finally a regulator with configuration $\mathtt{REG}_n(\mathcal{F}, a)$. Denote by d [resp., D] a lower bound [resp., an upper bound] for the delay of the non-lost data units of $\mathcal{F}$ through the system $\mathcal{S}$ between a and the output of the PEFs.

- *If $\mathcal{S}$ is lossless for $\mathcal{F}$ (*i.e.*, for every data unit m of the aggregate, at least one packet containing m reaches the PEFs), then d [resp., D] is also a lower bound [resp., an upper bound] for the delay of the non-lost data units through $\mathcal{S}'$, which we note $[d', D'] = [d, D]$.*
- *Otherwise, d [resp., $D + T^{POF}$] is a lower bound [resp., an upper bound] for the delay of the data units through $\mathcal{S}'$ with T^{POF} the timeout value of the POF.*

To prove the result, we simply observe that the POF in Figure 4.23 does not increase the delay bounds and $\mathcal{S}^\dagger$ is a FIFO system, thus the shaping-for-free property of regulators holds. The formal proof is in Appendix B.2.11. Therefore, the "PEF + POF + REG" configuration

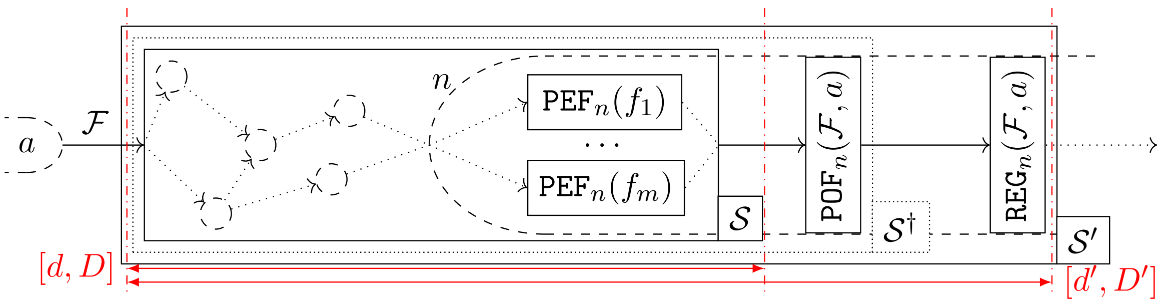


Figure 4.23: Notations of Theorem 4.6. An aggregate re-sequencing followed by a REG is placed after the PEFs. We are interested in the delay bounds through system $\mathcal{S}'$.

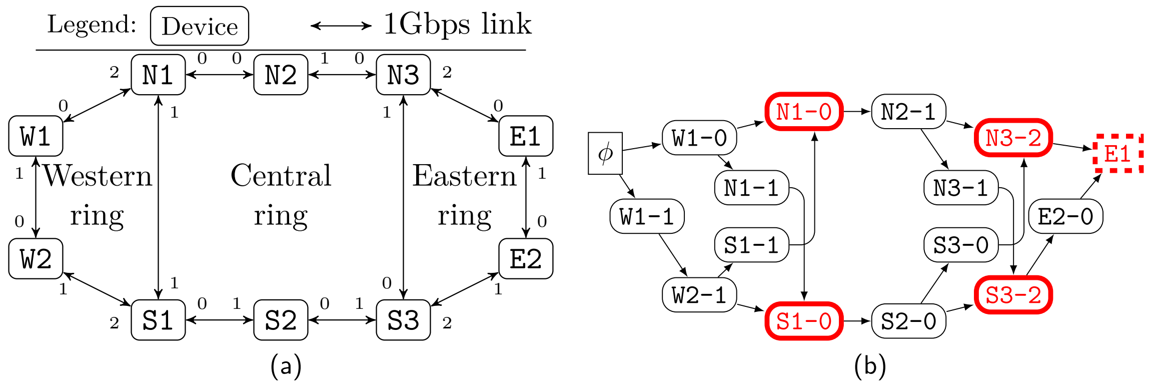


Figure 4.24: Parametric network for the evaluation of the delay bounds obtained with the *intuitive approach* and with xTFA. (a) Topology: a central ring with N_i devices connects two rings with N_e devices by using two gateways for each external ring. Here, $N_i = 6$ and $N_e = 4$. The numbers on the arrow tips distinguish the output ports of a device. (b) Graph of the flow W1 $\rightarrow$ E1 on the network of Fig. 4.24a. The vertex ϕ represents the source application. DEV-j represents output port j of device DEV. Output ports containing a PEF for the flow have a thick red outline. Destination is shown for convenience but not present in the actual flow graph.

provides all the benefits on the network performance bounds associated with the "PEF + POF" and the "PEF + REG" configurations, removing most of their drawbacks. This is summarized on the last line of Table 4.3. Only the hardware cost remains a drawback, as the models of Figures 4.8 and 4.9 must be implemented.

4.5 Evaluation of the Performance of The Toolbox

The xTFA tool (Chapter 6) implements the results of the toolbox of Section 4.3. It also takes into account the line-shaping effect (for details, we refer to Chapter 6). In this section, we compare the performance bounds obtained with xTFA with those obtained with an algorithm that relies on the *intuitive approach* that assumes the PEF is lossless but also models the line-shaping effect (as with FP-TFA in Chapter 3).

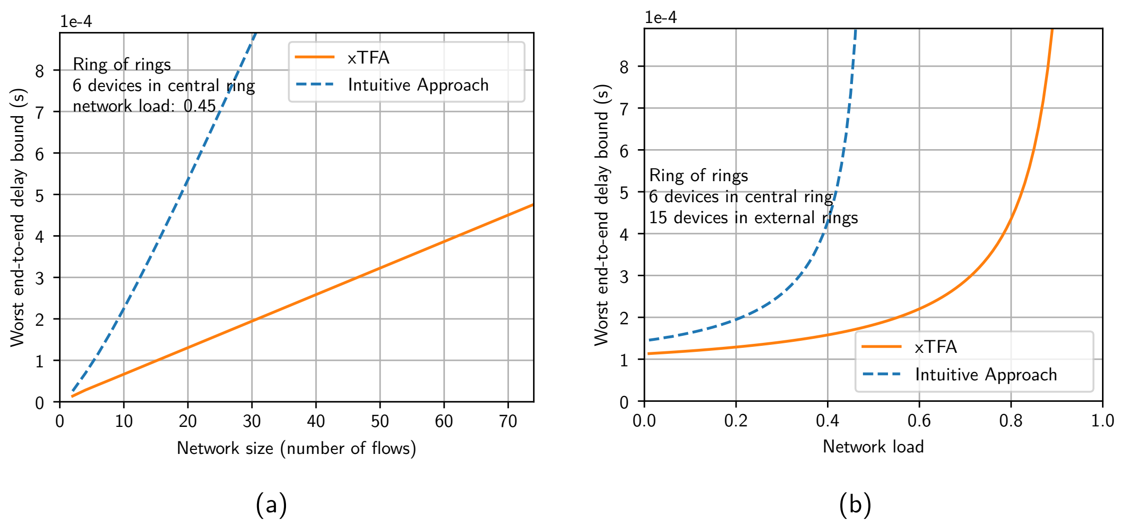


Figure 4.25: Comparison of the end-to-end delay bounds obtained on the parametric topology with each method. Lower is better. (a) As a function of the network size. (b) As a function of the network load.

4.5.1 Description of the Evaluation

We consider a parametric network inspired by [Heise, *et al.* 2014; Kirrmann, *et al.* 2009]. The physical topology is shown in Fig. 4.24a. We denote by N_e the number of devices on the external rings and by N_i the number of devices on the central ring. Devices are connected using full-duplex transmission links with capacity 1Gbps. Any device on the external rings is both a switch that interconnects two links and an end system that runs internal applications.

For each device on the external rings, we consider a unicast flow from this device to its symmetrical peer on the remote external ring. All flows have the same periodic profile at their source application, with a payload of 1200 bytes. Finally, all flows are redounded using PREFs, the four gateways as well as the destination eliminate the duplicates.

This is illustrated in Figure 4.24b that represents the flow graph $\mathcal{G}(f)$ of the flow f between `W1` and `E1`. The box with the special tag ϕ represents the source application that generates the data units. These data units are replicated and exit their source device `W1` through two output ports: `W1-0` and `W1-1`. Output ports containing a PEF for the flow `W1` $\rightarrow$ `E1` are marked with a thick red outline (the destination also removes the duplicates).

We consider a unique class of traffic. Each CBQS within the output ports offers to the aggregate a service rate equal to the capacity of the transmission link (1Gbps), and a latency of $2\mu s$. The resulting network is not feed-forward.

4.5.2 Comparison of the Delay Bounds when Varying the Network Size

We select $N_i = 6$ and vary N_e, the number of external nodes, from 3 to 39, which in turn varies the number of flows from 2 to 74. For any network size, we select the period for all flows such that the network load equals 45%. Figure 4.25a presents the ETE delay bounds obtained with each method as a function of the number of flows in the network.

We observe that xTFA provides the smallest delay bounds. The delay bounds obtained with the *intuitive approach* increase rapidly with the number of flows. This can easily be explained as the number of duplicates increase with the number of flows. These duplicates are assumed to not be removed by PREFs in the *intuitive approach*, which increases the apparent burst and rate of the aggregates.

4.5.3 Comparison of the Delay Bounds when Varying the Network Load

We now fix $N_i = 6$ and $N_e = 15$ (26 flows in the network) and vary the period of the flows to increase the network load from 1% to 99%. The obtained ETE delay bounds are shown in Fig. 4.25b. We observe that for each method, the obtained delay bound diverges when the network load increases. Moreover, the critical load u_{crit} (at which the computed bounds diverge) equals 0.89 with xTFA but only 0.49 with the *intuitive approach*.

Here again, the duplicates that are assumed not to be removed with the *intuitive approach* worsens drastically the burstiness cascade that generates the cyclic dependencies, and the implementation based on the *intuitive approach* struggles to find a finite fixed-point $(\overline{\boldsymbol{b}}, \overline{\boldsymbol{d}})$ to the $\mathcal{FF}$ application (Section 4.3.3) as the network load approaches 50%.

Conclusion

In this chapter, we have considered the side-effects of using redundancy mechanisms on latency bounds. We have first identified in Section 4.1 that the existing network-calculus literature does not permit the computation of latency bounds when the packet replication and elimination functions (PREFs) are implemented at intermediate nodes inside the network, whereas such a configuration is supported by the IEEE TSN and IETF DetNet standards. In fact, we have identified several challenges associated with packet-elimination functions (PEFs) placed at intermediate nodes. We have discussed an intuitive approach for tackling the challenges; it assumes that the PEF never eliminates any packets.

Then we have proposed a framework for modeling PREFs in network calculus. The framework is composed of a model for the redounded flows and for the functions (Section 4.2), a toolbox of network-calculus results that capture the effect of PREFs on the arrival curves (Theorem 4.1) and on the reordering metrics (Theorem 4.2). We also extend the fixed-point result (Theorem 4.3) to allow for the computation of end-to-end latency bounds in networks that contain both PREFs and cyclic dependencies.

Then we have analyzed the interactions between PREFs and traffic regulators (PFRs and IRs). Traffic regulators can be seen as an opportunity for canceling the burstiness increase caused by the redundancy, making the latter transparent to the nodes located after the PEF. However, when such a regulator is placed immediately after the PEF, it incurs a delay penalty because of the packet mis-ordering created by the redundancy. With PFRs, we were able to bound this delay penalty, but with IRs, we have exhibited a situation in which the IR generates unbounded latencies. Placing a packet-ordering function (POF) after the PEF and before the regulator solves the issue, but for the IR, a per-flow reordering is not sufficient and the POF must re-order the entire aggregate.

Finally we have compared our tight model of the PREFs with the intuitive approach on a parametric topology. We have observed that our tight model provides significantly tighter latency bounds than the intuitive approach. In particular, our xTFA tool (that relies on our tight model) can compute latency bounds for much larger load values than the highest possible network load of the intuitive approach.

In the next chapter we evaluate the side-effects of the third set of mechanisms identified in the thesis' Introduction: the time-synchronization mechanisms. In particular, we develop a framework for modeling clock non-idealities in network calculus.

Chapter 5

Effects of Clock Non-Idealities

Au moment où j'écris ces lignes, l'heure sonne à une horloge voisine ; mais mon oreille distraite ne s'en aperçoit que lorsque plusieurs coups se sont déjà fait entendre ; je ne les ai donc pas comptés. Et néanmoins, il me suffit d'un effort d'attention rétrospective pour faire la somme des quatre coups déjà sonnés, et les ajouter à ceux que j'entends. [...] Je m'aperçois que les quatre premiers sons avaient frappé mon oreille et même ému ma conscience [...] de manière à [...] faire une espèce de phrase musicale. Pour évaluer rétrospectivement le nombre des coups sonnés, j'ai essayé de reconstituer cette phrase par la pensée ; mon imagination a frappé un coup, puis deux, puis trois, et tant qu'elle n'est pas arrivée au nombre exact quatre, la sensibilité, consultée, a répondu que l'effet total différait qualitativement. [...] Bref, le nombre des coups frappés a été perçu comme qualité, et non comme quantité [...].

Henri Bergson, *Essai sur les données immédiates de la conscience.*

Contents

5.1 Related Work 95

5.2 Time Model 96

5.2.1 General Time Model 96

5.2.2 The Non-Synchronized Time Model 97

5.2.3 The Synchronized Time Model 100

5.3 Network-Calculus Toolbox for Networks With Non-Ideal Clocks . . . 100

5.3.1 Results on Delays 101

5.3.2 Results on Arrival Curves 101

5.3.3 Results on Service Curves 103

5.4 Computing End-To-End TAI Bounds in Time-Sensitive Networks . . 104

5.4.1 Two Opposite Strategies 104

5.4.2 Networks with Non-ideal Clocks and packet replication, elimination and ordering functions (PREOFs) 106

5.5 Regulators in Networks With Non-Ideal Clocks 107

5.5.1 Simplified Network Model For The Analysis of Regulators in Networks with Non-ideal Clocks 108

5.5.2 Non-synchronized Networks with Regulators 109

5.5.3 Synchronized Networks with Regulators 112

5.6 Evaluation of the Approaches 114

5.6.1 Comparison of the Two End-to-end Strategies 114
5.6.2 Comparison of the Regulator Adaptation Methods 115

In the previous chapters we have assumed that the notion of time is universal. It corresponds to assuming that the clock of each flow source and of each network element is perfectly aligned with a reference defining the "true time". In reality, the clocks used in a time-sensitive network are non-ideal and deviate slightly from the true time.

Time-sensitive networks are either synchronized or non-synchronized. In non-synchronized networks, local clocks run independently at every node and their deviations are not controlled. In synchronized networks, the deviations are kept within bounds, using a time-synchronization protocol or a global navigation satellite system. Tightly synchronized networks (with a precision of $\sim 1\mu s$) are typically analyzed as if clocks would be ideal. Other networks require synchronization only for management purposes and are synchronized with a precision of around $\sim 100ms$; we call them "loosely synchronized".

Among the network elements that we study in the previous chapters, the traffic regulators have shown to be powerful tools that can solve the issue of cyclic dependencies (Chapter 3) and correct for the increased burstiness due to the redundancy mechanisms (Chapter 4). But the traffic regulators (in particular, the IRs) are very susceptible to the packet mis-ordering, as outlined in Chapter 4. Similarly, they could be very susceptible to clock non-idealities.

Indeed, a regulator $\texttt{REG}_n(\mathcal{F}, o^*)$ that is configured for a flow $f \in \mathcal{F}$ with a rate r and a burst b (Figure 4.9) makes sure that over any window of *duration* t, no more than $rt + b$ bits of f are released by the regulator. Hence the evaluation of elapsed time is at the heart of the operation of any regulator. If the regulator relies an ideal clock and if the system $(o \to n)$ is FIFO for $\mathcal{F}$, then it enjoys the "shaping-for-free" property discussed in Chapter 3. But if the regulator bases its computations on a non-ideal clock, then the true rate enforced by $\texttt{REG}_n(\mathcal{F}, o^*)$ might differ from the configured rate. For example, if the configuration of the regulator is *non-adapted*, *i.e.*, does not take into account the clock non-idealities, and if the clock of the regulator is too slow, then the true rate enforced by the regulator is slightly less than the rate the flow had at o. With a greedy source, this will lead to a slow, but steady, buildup of backlog with arbitrarily large delay or unexpected loss.

This simple example suggests that clock non-idealities might significantly affect the delay analysis of time-sensitive networks with regulators. In this chapter, we determine to what extend this holds, in non-synchronized and synchronized networks, and if required, we propose some fixes. We also provide a framework for applying the network-calculus theory to networks with non-ideal clocks. This framework is of interest beyond the analysis of time-sensitive networks and of regulators. It is therefore independent from the model of time-sensitive networks. Specifically, our contributions are:

- We propose a time model for non-synchronized and synchronized networks; it can be used for computation of latency bounds.
- To compute latency bounds when clocks are non-ideal, we propose a toolbox to be used with other network calculus results.
- In non-synchronized networks, we show that the configuration of regulators must be adapted to take into account the clock non-idealities. If not adapted, we prove that regulators can yield unbounded latencies or unexpected packet losses.

• For non-synchronized networks, we propose and analyze two methods for configuring the regulators and for avoiding the problem mentioned above. One method, rate and burst cascade, requires that the parameters of a regulator depend on the position of the regulator along the flow path, thus it adds complexity to the control plane. It applies to both PFR and IR. The second method, asynchronous dual arrival-curve method (ADAM), uses the same regulator parameters at all re-shaping points on the flow path thus makes the control plane simpler; it applies to PFRs only.

• In synchronized networks, we compute a bound on the delay penalty imposed by per-flow regulators (PFRs). In tightly synchronized networks, this penalty is small compared to latency bounds, and the current practice of ignoring it is adequate. In contrast, in loosely synchronized networks, we show that the delay penalty can be significant.

• The conclusions are very different in synchronized networks with interleaved regulators (IRs). We show that, even in tightly synchronized networks, IRs can yield unbounded delay or unexpected loss if the residual clock inaccuracies are not accounted for. The method of rate and burst cascade can be used to avoid this problem.

The current chapter is organized as follows. In Sections 5.1, we present the related work. We then introduce our new time model in Section 5.2. We then detail our toolbox of network-calculus results in Section 5.3. In Section 5.4 we discuss how to apply the toolbox to compute end-to-end TAI delay bounds in time-sensitive networks. We last analyze regulators in non-synchronized and synchronized networks in Section 5.5. Part of the material presented in this chapter was published in [Thomas, Le Boudec 2020].

5.1 Related Work

The modeling of clock non-idealities benefits from a solid background in time metrology. In [ITU G.810], the International Telecommunication Union (ITU) defines fundamental notions and models for clocks used in synchronization networks. These models are further detailed in reference documents such as [ITU G.812] for the ITU and [IEEE 1139] for the IEEE. Clocks used in TSN networks shall comply with [IEEE 802.1AS, §B.1].

Many technologies have been developed to perform the time-synchronization of a network. The most common are the use of an external time source such as a global navigation satellite system (GNSS) [Powers, Hahn 2004], and the use of time-synchronization protocols such as Network Time Protocol (NTP) [RFC 5905], Precision Time Protocol (PTP) [IEEE 1588], generalized PTP (gPTP) [IEEE 802.1AS] and WhiteRabbit [Moreira, *et al.* 2009]. Each comes with various performance analyses: We can cite [Murta, Torres Jr. Mohapatra 2006] for NTP, [Dierikx, *et al.* 2016] for WhiteRabbit. The design of a “good” time-synchronization protocol remains an open issue [Freris, Graham, Kumar 2011; Ridoux, Veitch 2010].

In each of the above analyses, the precision of the time-synchronization protocol depends on the latency and jitter of synchronization messages and of control data. The latency and jitter bounds of time-sensitive networks were studied in numerous occasions using network calculus, as reported in Section 2.5 of Chapter 2.

Interestingly, the reciprocal aspect, *i.e.*, the effect of the clock and synchronization non-idealities on the network performances, appears to be much less studied. For example, we note that many performance analyses of IEEE TSN assume that the synchronization is perfect [Nasrallah, *et al.* 2019]. Even the famous network simulator ns-3 [16] has a unique time-base for simulating network events. In Section 7.2 of Chapter 7 we discuss the limited literature on simulating clock non-idealities in discrete-event simulators (DESs).

A seminal work on applying network calculus on networks with non-ideal clocks for obtaining worst-case upper-bounds lies in [Daigmorte, Boyer 2016; Daigmorte, Boyer 2017; Daigmorte, Boyer, Migge 2017]. In this set of papers, the authors are interested in a bandwidth-management method that spreads the time at which frames are scheduled on a CAN bus. The author note that the technique requires a time-synchronization mechanism between the network nodes. However, they show that a "weak synchronization" of the nodes (with a 1ms synchronization precision) already provides significant performance improvements [Daigmorte, Boyer 2016, §5.3]. Their time-model is limited to synchronized networks (including loosely synchronized networks). It does not consider any bound on the clock frequency offset, but only on the clock time-error bound (called the "phase bound" in their papers). In the thesis, we are interested in both synchronized and non-synchronized networks, and we show in this chapter that taking into consideration the bounds on the frequency offsets helps deriving tight delay bounds. Furthermore, the system model considered by [Daigmorte, Boyer 2016] is limited to specific arrival and service curves, whereas we are interested in computing the effect of clock non-idealities given any service curve, arrival curve or latency bound.

Industrial partners have also expressed their concerns on the effect of clock non-idealities in IEEE TSN. The recently-published amendment [IEEE 802.1Qcr] mentions the possible consequences of clock non-idealities when deploying asynchronous traffic shaping (ATS), the TSN implementation of the IR. The appendix [IEEE 802.1Qcr, §V.8] and proposes some solutions that would benefit from theoretical foundations, as proposed in this chapter.

5.2 Time Model

We propose a framework for modeling a clock within a device. We then derive this framework for non-synchronized networks and for synchronized networks. This model and the following Section 5.3 are independent from the TSN output-port model of Section 2.6.

5.2.1 General Time Model

We denote with $\mathcal{H}_{\text{TAI}}$ the true time, *i.e.*, the international atomic time (*temps atomique international*) (TAI). We assume that it represents a continuous quantity. When reading the time indicated by a clock $\mathcal{H}_i$ in the network, only a subset of values are readable, due to the precision of the clock logic. We assume that this clock logic enforces the accessible values to increase when the clock is read over the course of the true time. The red curve in Figure 5.1 represents the accessible values of the clock $\mathcal{H}_i$ as a function of the true time. It is possible to find a continuous, strictly increasing function h_i of the true time that returns the value that the clock $\mathcal{H}_i$ would display at true time t if it had infinite precision (Figure 5.1). Accessing the values of clock $\mathcal{H}_i$ corresponds to sampling the function h_i at the discrete time instants where the clock logic does a transition.

The origin of time is a relative notion, hence we assume that h_i also extends to $\mathbb{R}-$. Furthermore, we assume that $\lim_{t\to-\infty} h_i(t) = -\infty$ and $\lim_{t\to+\infty} h_i(t) = +\infty$. Hence, h_i is a permutation of $\mathbb{R}$ and for any $t \in \mathbb{R}$ both $h_i(t)$ and $h_i^{-1}(t)$ exist. This assumption excludes

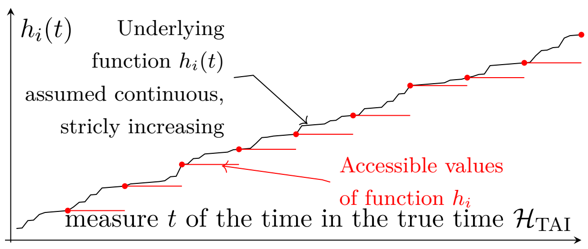

$h_i(t)$
Underlying function $h_i(t)$ assumed continuous, stricly increasing
Accessible values of function h_i
measure t of the time in the true time $\mathcal{H}_{\mathrm{TAI}}$

Figure 5.1: Time function h_i between the TAI and the clock time. Only a subset of values are readable. We assume that the underlying function h_i is continuous, strictly increasing.

some models of clocks (e.g., those whose frequency tends to zero as the true time increases), but it holds in the non-synchronized and synchronized time model introduced in Sections 5.2.2 and 5.2.3. It can also hold for the clocks whose frequency decreases (for instance due to the clock aging) but for which this decrease over the lifetime of the network is small with respect to the other non-idealities. A discussion on the clock aging is provided in Section 5.2.2. Function h_i allows us to introduce the relative time function, which will be useful in Section 5.3.

Definition 5.1 (Relative time function $d_{i\leftarrow g}$) *For two clocks $\mathcal{H}_g, \mathcal{H}_i$ we define the relative time function from $\mathcal{H}_g$ to $\mathcal{H}_i$ as $d_{i\leftarrow g} \triangleq h_i \circ h_g^{-1}$ where $\circ$ represents the composition.*

$d_{i\leftarrow g}(t)$ is the value that clock $\mathcal{H}_i$ would have when clock $\mathcal{H}_g$ shows time t if they both had infinite precision. By properties of h_i, h_g, $d_{i\leftarrow g}$ is a continuous, strictly increasing permutation of $\mathbb{R}$. Its inverse $d_{i\leftarrow g}^{-1}$ is equal to $d_{g\leftarrow i}$. The time-error function [ITU G.810, §4.5.13], between $\mathcal{H}_i$ and $\mathcal{H}_g$ is the function $t \mapsto d_{i\leftarrow g}(t) - t$ for any t measured with $\mathcal{H}_g$.

For two clocks $\mathcal{H}_g, \mathcal{H}_i$, $d_{i\leftarrow g}(0)$ is the value of the clock $\mathcal{H}_i$ when the clock $\mathcal{H}_g$ shows zero. We denote by T_{start} the maximum of this value for any pair $(\mathcal{H}_g, \mathcal{H}_i)$, where each of $\mathcal{H}_g, \mathcal{H}_i$ represents a clock in the network or the TAI. We assume that, for any pair of clocks, no device transmits any bit in the network until $d_{i\leftarrow g}$ reaches T_{start}. This is not a limiting assumption because we can assume that all devices start with a rough estimation of the true time, hence of each other. Consequently, T_{start} could be in the magnitude of hours or days, whereas the origin of time on network devices usually refers to several years in the past.

In this thesis, we consider two time-models:

- The non-synchronized time-model: Clocks are free-running and do not interact with each other, but constraints on their stability can be formulated.
- The synchronized time-model: In addition to the stability requirements on the clocks, a time-synchronization algorithm (such as NTP or PTP), or an external time source (such as a GNSS) is employed. It distributes a time reference to all devices so that their local time matches with each other within a specified bound.

5.2.2 The Non-Synchronized Time Model

We consider a clock $\mathcal{H}_i$ in the network. We first assume that this clock does not interact with any other. We assume that, when compared to the true time $\mathcal{H}_{\mathrm{TAI}}$, the clock behaves as per the time-error model provided in [ITU G.810, §I.3], reported below:

$$\forall t, \quad h_i(t) - t = x_{0,i} + t y_{0,i}(T) + \frac{D_i}{2} t^2 + \psi_i(t) \tag{5.1}$$

With $x_{0,i}$ the initial time offset of $\mathcal{H}_i$ (relative to the true time, in seconds), $y_{0,i}(T)$ the frequency offset at constant temperature T, defined relative to the true frequency of 1 second per second (thus with no unit), D_i the average frequency drift of clock $\mathcal{H}_i$ caused by its aging (in seconds^{-1}), and ψ_i is a random noise component (in seconds).

$x_{0,i}$ can take any value. Hence, we cannot constrain the time-error function itself but we can constrain its evolution. Take $s < t$, then (5.1) gives

$$h_i(t) - h_i(s) - (t-s) = (t-s)y_{0,i}(T) + \frac{D_i}{2}(t^2 - s^2) + \psi_i(t) - \psi_i(s) \tag{5.2}$$

• The first term is linear with $(t-s)$ and depends on $y_{0,i}(T)$. In industrial requirements, the frequency offset is usually bounded by a value that depends on the temperature. We note $y_{\max,i}(T)$ the bound on the frequency offset of clock $\mathcal{H}_i$ at constant temperature T and $\rho_{1,i} = \max_{\{T\in\mathcal{T}\}} y_{\max,i}(T)$ its highest value over the whole range of temperatures $T \in \mathcal{T}$ that the network is expected to encounter. It corresponds to $a_1 + a_2$ in [ITU G.812, §11.2.1].

• The second term is quadratic with $(t-s)$ and depends on the relative aging of the clocks.

In industrial requirements, such as TSN [IEEE 802.1AS, §B.1], the term D_i is often neglected. To validate the assumption that the aging is negligible, we compare the linear coefficient due to the frequency offset, $\rho_{1,i}$, with the "linear" coefficient due to the aging, $D_i\frac{t+s}{2}$. The following numerical application verify that we can neglect D_i.

Numerical application to IEEE TSN: Call L the order of magnitude of the lifetime of a network (in seconds). Then the aging coefficient over L is bounded by $D_i \cdot L$ (no unit). If we take for $\rho_{1,i}$ the value specified in [IEEE 802.1AS, §B.1.1] and for D_i (not specified in TSN) the largest value specified in [ITU G.812, Table 24], we obtain $L = \rho_{1,i}/D_i \approx 10^{-4}/10^{-14} = 10^{10}s$. For the aging to be noticeable compared to the acceptable frequency offset, the network shall be in operation for more than 300years.

• The last term of Equation (5.2) is made of noise and is further detailed in [ITU G.812, §8]. It has two components. The former is the timing jitter [ITU G.810, §4.1.12], a high-frequency signal. It is constrained by a peak-to-peak jitter bound η_i [IEEE 802.1AS, §B.1.3.1].

Numerical application to IEEE TSN: In [IEEE 802.1AS, §B.1.3.1], the jitter of any clock $\mathcal{H}_i$ shall not exceed 2ns peak-to-peak, that is $\eta_i = $ 2ns for any $\mathcal{H}_i$ in the network.

• The last noise component, the wander [ITU G.810, §4.1.15], is a low-frequency noise signal. As opposed to the jitter or to the frequency offset, it is usually constrained by using the time deviation (TDEV), a stochastic metric (or by using the Allan Deviation, ADEV). The TDEV of clock $\mathcal{H}_i$, $\text{dev}_i(t-s)$ is an upper-bound on the deviation of the time-error function over an observation period $t-s$. We assume that we can find $m_i \in \mathbb{R}$ such that the probability of the time-error function to present a wander over $(t-s)$ higher than $m_i \cdot \text{dev}_i(t-s)$ is negligible over the lifetime of the network.

In the majority of technical requirements, as in TSN [IEEE 802.1AS, Annex B.1], the TDEV is in the form $\text{dev}_i(t-s) = (t-s)c_i$, with c_i a constant. In some cases, it can even be or negligible [ITU G.812, §8]. To remain conservative, we consider the linear form and we define $\rho_{2,i} = m_i \cdot c_i$. Hence, the wander of Equation (5.2) is upper-bounded by $\rho_{2,i} \cdot (t-s)$.

We define the stability of clock $\mathcal{H}_i$ as $\rho_i = 1 + \rho_{1,i} + \rho_{2,i}$. The linear coefficient of Equation (5.2) is bounded by $\rho_i - 1$. In general, one of the two terms $\rho_{1,i}$, $\rho_{2,i}$ is negligible with respect to the other. For example, when the non-synchronized clock $\mathcal{H}_i$ uses the phase-locking mechanism [ITU G.810, §4.4.4] (also called syntonization or frequency synchronization [ITU G.8261]) with $\mathcal{H}_{\text{TAI}}$, then $\rho_{1,i}$ equals zero [ITU G.810, §I.3]. When phase-locking mechanisms are not used (as in IEEE TSN), then $\rho_{1,i}$ is usually much higher than c_i thus than $\rho_{2,i}$.

Numerical application to IEEE TSN: In the TSN requirements, for any clock $\mathcal{H}_i$ we have: $\rho_{1,i} = 100ppm$ [IEEE 802.1AS, §N.1.1] and $c_i \leq 5 \cdot 10^{-9}$ [IEEE 802.1AS, Table B.1]. Hence, even with a margin m_i of hundred times the standard deviation, $\rho_{2,i}$ remains much smaller than $\rho_{1,i}$ and we obtain that any clock $\mathcal{H}_i$ has the stability $\rho_i = 1+1\cdot 10^{-4}$.

From the above considerations, Equation (5.2) has (1) a jitter, high-frequency term, constrained by η_i, (2) a linear term, bounded by $\rho_i - 1$, and (3) no higher-level terms. Hence:

$$\forall t \geq s, \quad h_i(t) - h_i(s) \leq (t-s)\rho_i + \ \eta_i \tag{5.3}$$

We now lower-bound the evolution by changing h_i for h_i^{-1} and we obtain:

$$\forall t \geq s, \quad (t-s-\eta_i)\frac{1}{\rho_i} \leq h_i(t) - h_i(s) \leq (t-s)\rho_i + \ \eta_i \tag{5.4}$$

Let us now consider a pair of clocks $(\mathcal{H}_i, \mathcal{H}_g)$. We obtain, for $t \geq s$:

$$\begin{aligned} d_{i\leftarrow g}(t) - d_{i\leftarrow g}(s) &= h_i(h_g^{-1}(t)) - h_i(h_g^{-1}(s)) \\ &\leq (t-s)\rho_i\rho_g + \eta_g\rho_i + \eta_i \end{aligned}$$

For a given network, we define the clock-stability bound of the network as $\rho = \max_{\{\mathcal{H}_i,\mathcal{H}_g\}}(\rho_i\rho_g)$ and the time-jitter bound of the network as $\eta = \max_{\{\mathcal{H}_i,\mathcal{H}_g\}}(\eta_g\rho_i + \eta_i)$. We have finally obtained the following model: for any pair $(\mathcal{H}_g, \mathcal{H}_i)$, $\forall t \geq s$,

$$\frac{1}{\rho}(t-s-\eta) \leq d(t) - d(s) \leq \rho(t-s) + \eta \tag{5.5}$$

where $d = d_{i\leftarrow g}$. Note that ρ and η do not depend on $\mathcal{H}_g, \mathcal{H}_i$. With $\mathcal{H}_g = \mathcal{H}_{\text{TAI}}$, Equation 5.5 proves that the assumption $[\lim_{t\to-\infty} h_i(t) = -\infty, \lim_{t\to+\infty} h_i(t) = +\infty]$ holds.

Figure 5.2a presents, for a given known starting point $(s, d(s))$, the possible evolution space of $d(t)$ in the non-synchronized model as well as a possible trajectory. We note that the time-error function $t \mapsto (d(t) - t)$ can be unbounded in this model.

Numerical application to IEEE TSN: Any clock $\mathcal{H}_i$ satisfies $\rho_i = 1 + 1\cdot 10^{-4}$ and $\eta_i =$ 2ns. Hence, for a TSN network, the clock-independent parameters are $\rho = 1 + 2\cdot 10^{-4}$ and $\eta =$ 4ns. Note that the above values are minimum requirements for clocks to be used as local clocks in an IEEE TSN network. Of course, if better bounds are known from the manufacturer specifications, the values of ρ and η can be updated accordingly.

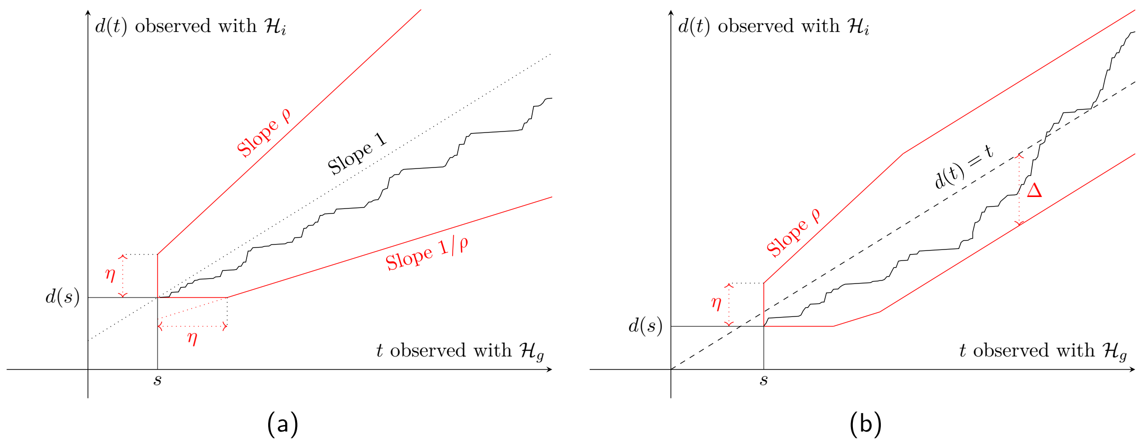

$d(t)$ observed with $\mathcal{H}_i$
Slope ρ
Slope 1
Slope $1/\rho$
η
η
$d(s)$
t observed with $\mathcal{H}_g$
s
(a)
$d(t)$ observed with $\mathcal{H}_i$
$d(t) = t$
Δ
Slope ρ
η
$d(s)$
t observed with $\mathcal{H}_g$
s
(b)

Figure 5.2: Envelope of $d(t)$ (red) and example of a possible evolution of $d(t)$ (black). (a) In the non-synchronized time-model. (b) In the synchronized time-model.

5.2.3 The Synchronized Time Model

In a synchronized network, the clocks that meet the above stability requirements are synchronized with each other. The synchronization can be performed using for example PTP [IEEE 1588], gPTP defined in TSN [IEEE 802.1AS], NTP [RFC 5905], WhiteRabbit [Moreira, *et al.* 2009] or a GNSS [Powers, Hahn 2004]. When synchronization is used, the time-error function for any pair such as $(\mathcal{H}_g, \mathcal{H}_i)$ is bounded by the precision of the protocol.

For a given network, we define $\Delta \geq 0$ the synchronization precision of the network and we assume that for any pair $(\mathcal{H}_g, \mathcal{H}_i)$, d meets the constraints of Equation (5.5), plus

$$\forall t, \quad |d(t) - t| \leq \Delta \tag{5.6}$$

where $d = d_{i \leftarrow g}$. Note that Δ does not depend on $\mathcal{H}_g, \mathcal{H}_i$.

Numerical application to IEEE TSN: For tightly synchronized networks, we take Δ =1μs from [IEEE 802.1AS, Normative Annex B.3]. For loosely synchronized networks, we select the precision of NTP. In [RFC 5905, Figure 27], NTP defines a “step threshold” of 125ms, and a survey of the NTP network notes that this value is hardly exceeded if the clock is synchronized [Murta, Torres Jr. Mohapatra 2006, §IV.B.1]. Hence, we take Δ =125ms.

Figure 5.2b presents, for a known starting point $(s, d(s))$, the possible evolution space of $d(t)$ in the synchronized model as well as a possible trajectory. Note that the Δ-large envelope is not centered on the starting point but on the $d(t) = t$ function.

5.3 Network-Calculus Toolbox for Networks With Non-Ideal Clocks

In the network-calculus framework (Chapter 2), the universality of the time notion is implicit. Hence, the results of Chapter 2 and in particular the three-bound theorem (Theorem 2.2) are

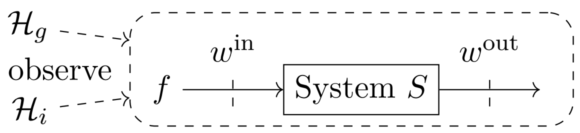


Figure 5.3: Clocks $\mathcal{H}_g$ and $\mathcal{H}_i$ observe flow f that enters system S.

valid whenever all the notions that appear in their formulation (the arrival curves, the service curves, the delays) are expressed (observed) using the same clock. Therefore, we propose here a toolbox that can be used to change the clock that observes one of the above notions. This results in an extension of network calculus to networks with non-ideal clocks.

In the entire section, we consider a system S and a flow f that crosses this system (see Figure 5.3). We denote by w^{in} [resp., w^{out}] the observation point located at the input [resp., at the output] of the system. We also consider two clocks $\mathcal{H}_i$ and $\mathcal{H}_g$.

5.3.1 Results on Delays

Proposition 5.1 (Changing the clock for the measure of a duration)
If $\chi^{\mathcal{H}_i}$ is the measure of a duration with clock $\mathcal{H}_i$, then the measure $\chi^{\mathcal{H}_g}$ of the same duration with $\mathcal{H}_g$ is bounded by:

$$\max\left(0, \frac{\chi^{\mathcal{H}_i}-\eta}{\rho}, \chi^{\mathcal{H}_i}-2\Delta\right) \quad\leq\quad \chi^{\mathcal{H}_g} \quad\leq\quad \min\left(\rho\chi^{\mathcal{H}_i}+\eta, \chi^{\mathcal{H}_i}+2\Delta\right) \tag{5.7}$$

with the convention that $\Delta = +\infty$ if the network is non-synchronized.

The proof is in Appendix B.3.1. A *delay* is a measure of the duration needed by a bit (or packet) to cross a system. If it is measured with $\mathcal{H}_{\text{TAI}}$, we call it a TAI delay.

If the delay of flow f through S in Figure 5.3 is lower-bounded by $d^{\mathcal{H}_i}_{f,S}$ [resp., upper-bounded by $D^{\mathcal{H}_i}_{f,S}$] when observed with $\mathcal{H}_i$, then by application of Proposition 5.1, its delay through S is lower-bounded by $d^{\mathcal{H}_g}_{f,S}$ [resp., upper-bounded by $D^{\mathcal{H}_g}_{f,S}$] when observed with $\mathcal{H}_g$, with

$$d^{\mathcal{H}_g}_{f,S} = \max\left(0, d^{\mathcal{H}_i}_{f,S}-\eta/\rho, d^{\mathcal{H}_i}_{f,S}-2\Delta\right) \qquad\text{and}\qquad D^{\mathcal{H}_g}_{f,S} = \min\left(\rho D^{\mathcal{H}_i}_{f,S}+\eta, D^{\mathcal{H}_i}_{f,S}+2\Delta\right) \tag{5.8}$$

Numerical application to IEEE TSN: For both time models, when a delay bound (either upper or lower bound) is within 1μsec and 200msec, the relative increase of the delay when observed with another clock ranges from 0.4% to 0.02%. Practically, the effect of clock non-idealities on the *definition* of the delay can thus be ignored.

5.3.2 Results on Arrival Curves

We denote by $R^{\mathcal{H}_i}_{f,w}$ the function such that $\forall t < 0, R^{\mathcal{H}_i}_{f,w}(t) = 0$ and $\forall t \geq 0, R^{\mathcal{H}_i}_{f,w}(t)$ is the number of bits of f that cross w between the instant measured as 0 using $\mathcal{H}_i$ and the instant measured as t using $\mathcal{H}_i$ (Figure 5.3). Then

Table 5.1: Relations Between an Arrival Curve Observed with $\mathcal{H}_i$ and an Arrival Curve Observed with $\mathcal{H}_g$.

	Arrival curve	
	General	Leaky-Bucket
in $\mathcal{H}_i$	$\alpha^{\mathcal{H}_i}(t)$	$\gamma_{r,b}$
in $\mathcal{H}_g$, general time model	$\alpha^{\mathcal{H}_i}((d_i \oslash d_i)(t))$	–
in $\mathcal{H}_g$, non-synchronized time model	$\alpha^{\mathcal{H}_i}(\rho t + \eta)$	$\gamma_{r\rho,b+r\eta}$
in $\mathcal{H}_g$, synchronized time model	$\alpha^{\mathcal{H}_i}(\min[\rho t + \eta, t + 2\Delta])$	$\gamma_{r\rho,b+r\eta} \wedge \gamma_{r,b+2r\Delta}$

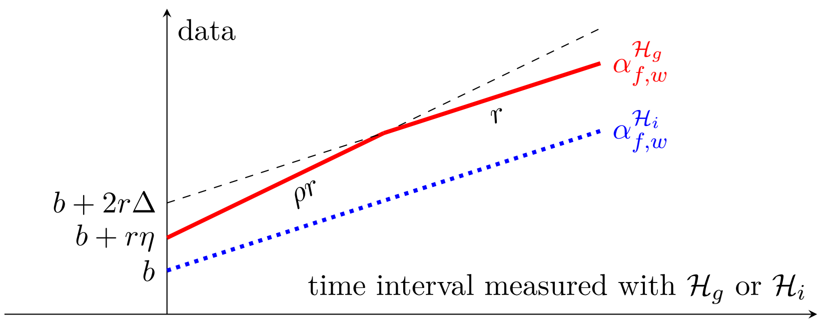


Figure 5.4: Changing the observing clock for a leaky-bucket arrival curve. A flow f has the leaky-bucket arrival curve $\alpha^{\mathcal{H}_i}_{f,w} = \gamma_{r,b}$ at w when it is viewed from $\mathcal{H}_i$. Then, f has the arrival curve $\alpha^{\mathcal{H}_g}_{f,w}$ when it is viewed from $\mathcal{H}_g$, with the convention $\Delta = +\infty$ if $\mathcal{H}_g$ and $\mathcal{H}_i$ are not synchronized.

Proposition 5.2 (Changing the clock of a cumulative function)
For any clock pair $(\mathcal{H}_g, \mathcal{H}_i)$, for any f any w, $\forall t \geq 0$, $R^{\mathcal{H}_g}_{f,w}(t) = R^{\mathcal{H}_i}_{f,w}(d_{i\leftarrow g}(t))$

The proof of the proposition is given in Appendix B.3.2. We now define the concept of an arrival curve observed with a clock.

Definition 5.2 (Arrival curve of a flow observed with a clock) *We say that a wide-sense increasing function α is an arrival curve for the flow f observed at w with clock $\mathcal{H}_i$ if $\forall t \geq s \geq 0$, $R^{\mathcal{H}_i}_{f,w}(t) - R^{\mathcal{H}_i}_{f,w}(s) \leq \alpha(t-s)$. We note such a function $\alpha^{\mathcal{H}_i}_{f,w}$.*

In particular, when $\mathcal{H}_i$ is $\mathcal{H}_{\text{TAI}}$, we retrieve Definition 2.5.

Proposition 5.3 (Changing the clock of an arrival curve)
If $\alpha^{\mathcal{H}_i}_{f,w}$ is an arrival curve for the flow f at w when observed with $\mathcal{H}_i$, then $\alpha^{\mathcal{H}_g}_{f,w} : t \mapsto \alpha^{\mathcal{H}_i}_{f,w}((d \oslash d)(t))$ is an arrival curve for the flow observed with $\mathcal{H}_g$, where $\oslash$ is the min-plus deconvolution (Definition 2.3) and $d = d_{i\leftarrow g}$.

The proof of the proposition is available in Appendix B.3.3. We can now find an arrival curve for a flow as observed from any clock as long as we know one that is observed with one clock. The application of Proposition 5.3 to the synchronized and non-synchronized time models is reported in Table 5.1. The table can be used for any pair of clocks $(\mathcal{H}_g, \mathcal{H}_i)$, each being either a clock in the network or the TAI.

In the right column, we apply it for a leaky-bucket curve. If $\alpha^{\mathcal{H}_i}_{f,w} = \gamma_{r,b}$ is a leaky-bucket arrival curve for f at w when observed with $\mathcal{H}_i$ (dotted blue curve in Figure 5.4), then the red curve $\alpha^{\mathcal{H}_g}_{f,w}$ in Figure 5.4 is an arrival curve for f at w when observed with $\mathcal{H}_g$. Here again we use the convention $\Delta = +\infty$ if the network is non-synchronized.

Table 5.2: Relations Between a Service Curve of a System Observed with $\mathcal{H}_i$ and a Service Curve of the same System Observed with $\mathcal{H}_g$.

	Service curve		
	General	Rate-Latency	Leaky-Bucket
in $\mathcal{H}_i$	$\beta^{\mathcal{H}_i}(t)$	$\lambda_{R,T}$	$\gamma_{r,b}$
in $\mathcal{H}_g$, general time model	$\beta^{\mathcal{H}_i}((d_i \overline{\oslash} d_i)(t))$	–	–
in $\mathcal{H}_g$, non-synchronized time model	$\beta^{\mathcal{H}_i}(1/\rho \cdot \lvert t-\eta \rvert^+)$	$\lambda_{R/\rho,\rho T+\eta}$	$\delta_\eta \otimes \gamma_{r/\rho,b}$
in $\mathcal{H}_g$, synchronized time model	$\beta^{\mathcal{H}_i}(\max[1/\rho \cdot (t-\eta), t-2\Delta, 0])$	$\lambda_{R/\rho,\rho T+\eta} \vee \lambda_{R,T+2\Delta}$	$(\delta_\eta \otimes \gamma_{r/\rho,b}) \vee (\delta_{2\Delta} \otimes \gamma_{r,b})$

Numerical application to IEEE TSN: For a non-synchronized network, changing the observing clock for a leaky-bucket worsens the rate by 0.02% and the burst by less than a bit for most flows (below 250Mbits/s) as η =4ns.

If the network is also synchronized, we obtain a second leaky-bucket arrival curve $\gamma_{r,b+2r\Delta}$ with an unchanged rate but with an increased burst. For a flow of 500kbits/s, this represents a burst increase of 125kbits for a loosely synchronized network and 1bit for a tightly synchronized network. For loosely synchronized networks, due to the high burst increase, the other part of the arrival curve, $\gamma_{\rho r,b+r\eta}$, needs to be used in order to obtain tight bounds.

5.3.3 Results on Service Curves

As for the arrival-curve concept, we define the service curve concept relative to a clock:

Definition 5.3 (Service-curve offered by a system to a flow when observed with a clock) *Consider a system S and denote by w^{in} and w^{out} its input and output observation points, respectively. We say that the system S offers to f the service-curve $\beta \in \mathfrak{F}_0$ when observed with $\mathcal{H}_i$ if $\forall t \geq 0, \quad R^{\mathcal{H}_i}_{f,w^{out}}(t) \geq (R^{\mathcal{H}_i}_{f,w^{in}} \otimes \beta)(t) \quad$ where $\otimes$ denotes the min-plus convolution (Definition 2.2). We note such a function $\beta^{\mathcal{H}_i}$.*

Proposition 5.4 (Changing the clock for a service curve) *If S offers the service-curve $\beta^{\mathcal{H}_i}$ when observed with $\mathcal{H}_i$, then it offers the service curve $\beta^{\mathcal{H}_g}$ when observed with $\mathcal{H}_g$ with $\beta^{\mathcal{H}_g} : t \mapsto \beta^{\mathcal{H}_i}((d \overline{\oslash} d)(t))$ where $d = d_{i \leftarrow g}$ and $d \overline{\oslash} d(t) = \inf_{u \geq 0}[d(t+u) - d(u)]$ denotes the max-plus de-convolution.*

The proof is in Appendix B.3.4. The application of Proposition 5.4 to the synchronized and non-synchronized time model is reported in Table 5.2. If S offers the rate-latency service curve $\beta^{\mathcal{H}_i}_S = \beta_{R,T}$ when observed with $\mathcal{H}_i$ (dotted blue curve in Figure 5.5a), then it offers the service curve $\beta^{\mathcal{H}_g}_S$ shown in red in Figure 5.5a when observed with $\mathcal{H}_g$. If S offers the leaky-bucket service curve $\beta^{\mathcal{H}_i}_S = \gamma_{r,b}$ when observed with $\mathcal{H}_i$ (dotted blue in Figure 5.5b, for example, S can be a greedy shaper with shaping curve $\gamma_{r,b}$, see Theorem 2.4), then it offers the service curve $\beta^{\mathcal{H}_g}_S$ shown in red in Figure 5.5b when observed with $\mathcal{H}_g$.

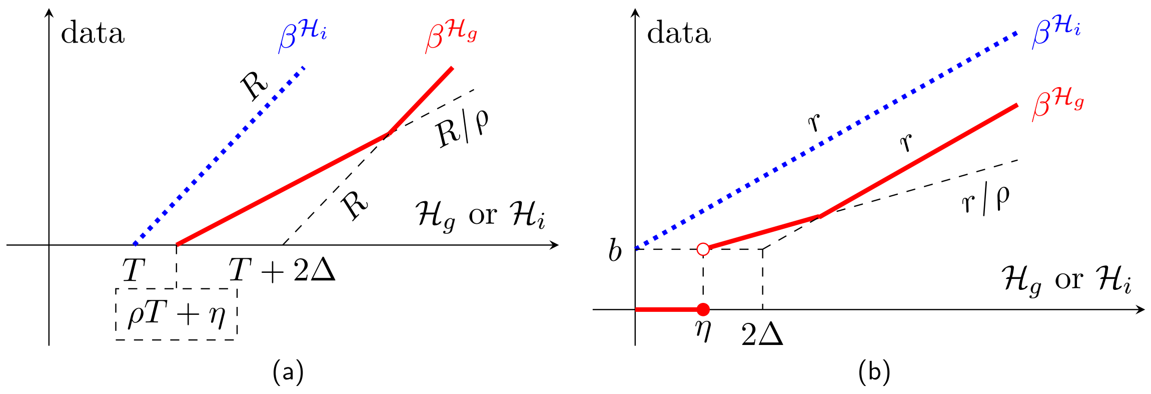


Figure 5.5: Changing the observing clock for rate-latency and leaky-bucket service curves. If the system offers service curve $\beta_{\mathcal{H}_i}$ when observed with $\mathcal{H}_i$, then it offers service curve $\beta_{\mathcal{H}_g}$ when observed with $\mathcal{H}_g$. (a) $\beta^{\mathcal{H}_i}$ is a rate-latency service curve. (b) $\beta^{\mathcal{H}_i}$ is the service curve of a PFR (leaky-bucket service curve).

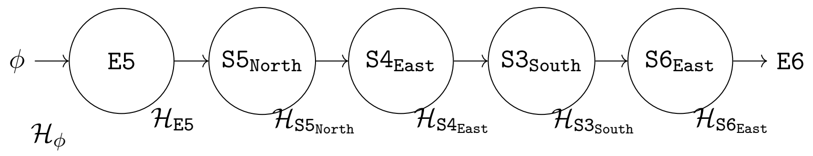


Figure 5.6: Flow graph for the flow of the toy example.

5.4 Computing End-To-End TAI Bounds in Time-Sensitive Networks

In this section, we discuss how the results of the toolbox of the previous section can be practically applied for computing end-to-end TAI latency bounds in time-sensitive networks, including those that contain packet replication, elimination and ordering functions (PREOFs).

5.4.1 Two Opposite Strategies

For general time-sensitive networks, two opposite strategies can be envisioned for computing end-to-end TAI latency bounds. To illustrate them, we consider the following example.

Example: Consider the non-redounded flow of the toy example of Figure 4.1, for which we provide the flow graph in Figure 5.6. We have seven different clocks (marked below the nodes). The flow meets is source arrival-curve constraint at ϕ, but only when observed with $\mathcal{H}_\phi$ the source of the clock. Similarly, the service offered by each node n is only guaranteed when observed with the clock $\mathcal{H}_n$ of the output port.

A first strategy for computing TAI end-to-end latency bounds is to translate all the source arrival curves and all the service curves in the TAI: we call this strategy "always in $\mathcal{H}_{\mathrm{TAI}}$".

Example: This strategy is illustrated in Figure 5.7. The source arrival curve is converted into an arrival curve observed with $\mathcal{H}_{\mathrm{TAI}}$ by using Proposition 5.3 (orange arrow). Then each service curve is converted into a service curve observed with $\mathcal{H}_{\mathrm{TAI}}$ using Proposition 5.4 (dashed red arrows).

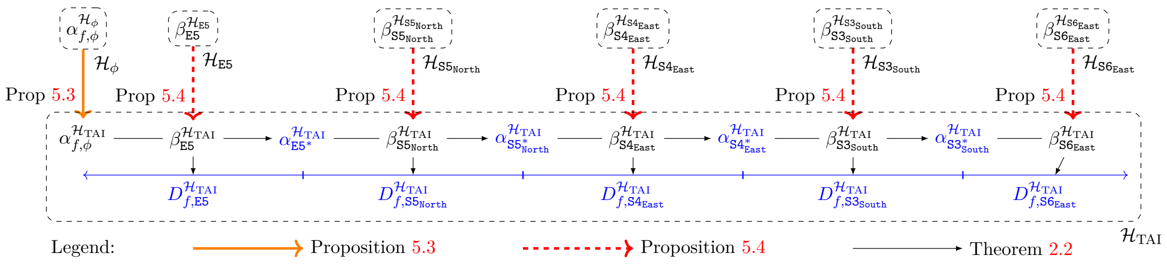

$\alpha_{f,\phi}^{\mathcal{H}_\phi}$
$\mathcal{H}_\phi$
Prop 5.3
$\beta_{\text{E5}}^{\mathcal{H}_{\text{E5}}}$
$\mathcal{H}_{\text{E5}}$
Prop 5.4
$\beta_{\text{S5}_{\text{North}}}^{\mathcal{H}_{\text{S5}_{\text{North}}}}$
$\mathcal{H}_{\text{S5}_{\text{North}}}$
Prop 5.4
$\beta_{\text{S4}_{\text{East}}}^{\mathcal{H}_{\text{S4}_{\text{East}}}}$
$\mathcal{H}_{\text{S4}_{\text{East}}}$
Prop 5.4
$\beta_{\text{S3}_{\text{South}}}^{\mathcal{H}_{\text{S3}_{\text{South}}}}$
$\mathcal{H}_{\text{S3}_{\text{South}}}$
Prop 5.4
$\beta_{\text{S6}_{\text{East}}}^{\mathcal{H}_{\text{S6}_{\text{East}}}}$
$\mathcal{H}_{\text{S6}_{\text{East}}}$
Prop 5.4
$\alpha_{f,\phi}^{\mathcal{H}_{\text{TAI}}}$
$\beta_{\text{E5}}^{\mathcal{H}_{\text{TAI}}}$
$\alpha_{\text{E5}^*}^{\mathcal{H}_{\text{TAI}}}$
$\beta_{\text{S5}_{\text{North}}}^{\mathcal{H}_{\text{TAI}}}$
$\alpha_{\text{S5}^*_{\text{North}}}^{\mathcal{H}_{\text{TAI}}}$
$\beta_{\text{S4}_{\text{East}}}^{\mathcal{H}_{\text{TAI}}}$
$\alpha_{\text{S4}^*_{\text{East}}}^{\mathcal{H}_{\text{TAI}}}$
$\beta_{\text{S3}_{\text{South}}}^{\mathcal{H}_{\text{TAI}}}$
$\alpha_{\text{S3}^*_{\text{South}}}^{\mathcal{H}_{\text{TAI}}}$
$\beta_{\text{S6}_{\text{East}}}^{\mathcal{H}_{\text{TAI}}}$
$D_{f,\text{E5}}^{\mathcal{H}_{\text{TAI}}}$
$D_{f,\text{S5}_{\text{North}}}^{\mathcal{H}_{\text{TAI}}}$
$D_{f,\text{S4}_{\text{East}}}^{\mathcal{H}_{\text{TAI}}}$
$D_{f,\text{S3}_{\text{South}}}^{\mathcal{H}_{\text{TAI}}}$
$D_{f,\text{S6}_{\text{East}}}^{\mathcal{H}_{\text{TAI}}}$
Legend:
Proposition 5.3
Proposition 5.4
Theorem 2.2
$\mathcal{H}_{\text{TAI}}$

Figure 5.7: Illustration of the strategy "everything in $\mathcal{H}_{\text{TAI}}$" for the example of Figure 5.6.

In $\mathcal{H}_{\text{TAI}}$ (dashed box on the lower-end of Figure 5.7), we obtain a source with an arrival curve $\alpha_{f,\phi}^{\mathcal{H}_{\text{TAI}}}$ and a set of successive service-curve elements with service curve $\beta_n^{\mathcal{H}_{\text{TAI}}}$. Therefore, remaining in this time domain, we can apply the network calculus results of Chapter 2 (black arrows on the lower-end of Figure 5.7).

All the intermediate results (in blue in Figure 5.7) are generated within $\mathcal{H}_{\text{TAI}}$. In particular, summing the per-hop delays directly provides the end-to-end latency bound for f, observed with $\mathcal{H}_{\text{TAI}}$.

The above strategy only uses Proposition 5.3 and 5.4, it does not use Proposition 5.1. This strategy is useful when the model of a node (observed with the node's clock) can be translated into a model observed with $\mathcal{H}_{\text{TAI}}$. Proposition 5.4 proves that anything that can be modeled as a service-curve element when observed with its local clock can be translated into a service-curve element with a different service curve when observed with $\mathcal{H}_{\text{TAI}}$.

An opposite strategy, that we denote as "always in local time" is to always apply the network-calculus results in the local clock of the output port.

Example: This strategy is illustrated in Figure 5.8. Here, the source arrival curve $\alpha_{f,\phi}^{\mathcal{H}_\phi}$ is first translated into an arrival curve as observed by the first output port $\alpha_{f,\phi}^{\mathcal{H}_{\text{E5}}}$.

Then for each node n, the network-calculus results (black arrows) are applied in the local clock $\mathcal{H}_n$. This provides the output arrival curve $\alpha_{f,n^*}^{\mathcal{H}_n}$ and the delay bound $D_{f,n}^{\mathcal{H}_n}$ observed with $\mathcal{H}_n$. Then the output arrival curve $\alpha_{f,n^*}^{\mathcal{H}_n}$ is translated into an output arrival curve $\alpha_{f,n^*}^{\mathcal{H}_{n+1}}$, but observed with the clock $\mathcal{H}_{n+1}$ of the next output port using Proposition 5.3 (orange arrows).

The per-hop delay bounds are only known so far when observed with the local clocks, hence they are translated to $\mathcal{H}_{\text{TAI}}$ using Proposition 5.4 (dashdotted purple arrows).

The above strategy only uses Propositions 5.3 and 5.1. This strategy is useful when the model of a node (observed with the node's clock) cannot be translated into a model observed with $\mathcal{H}_{\text{TAI}}$. The next subsection provides an example.

In the above example, the "always in local time" strategy requires ten calls to the toolbox[1], whereas the "always in TAI" strategy only requires six calls to the toolbox[2]. Each time a result of the toolbox in Section 5.3 is applied, the arrival curves are expanded and the service curves are weakened. Therefore, we prefer the strategy "always in TAI" whenever we process service-curve elements. Intermediate strategies mixing the above two can also be envisioned

[1]Five calls to Proposition 5.3 and five calls to Proposition 5.1
[2]One call to Proposition 5.3 and five calls to Proposition 5.4

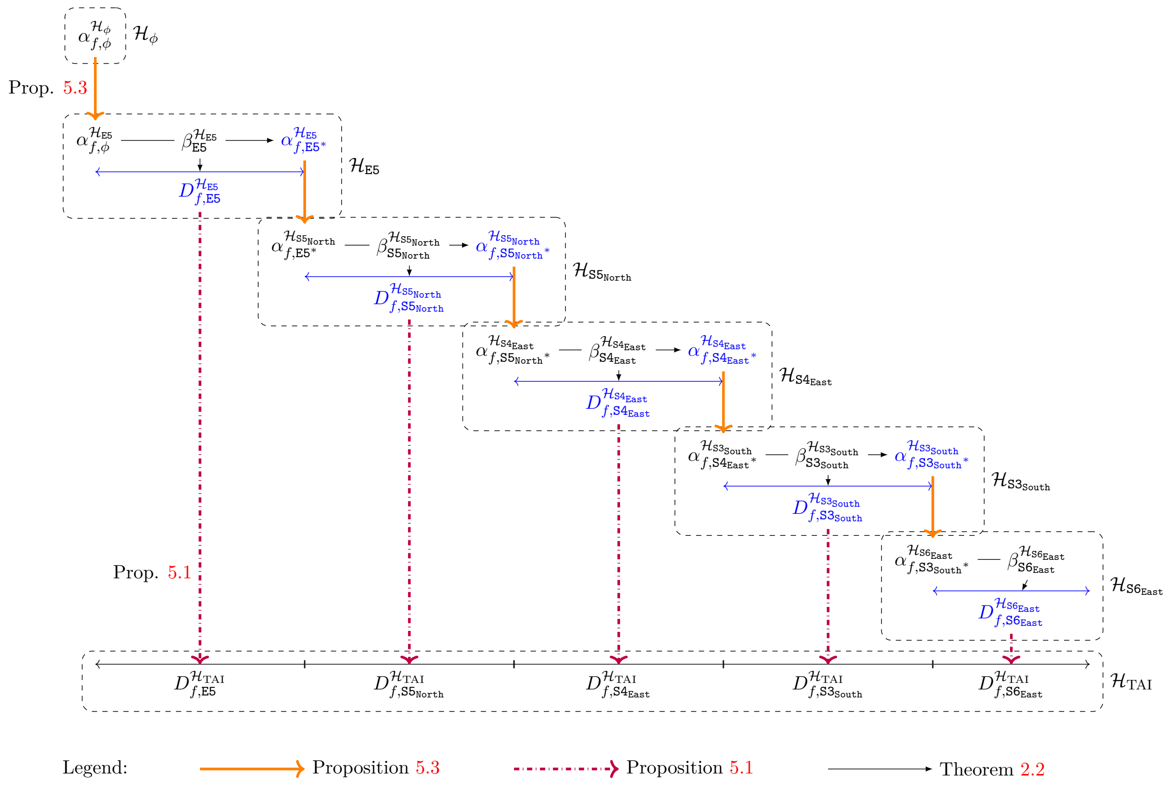

$\alpha^{\mathcal{H}_\phi}_{f,\phi}$
$\mathcal{H}_\phi$
Prop. 5.3
$\alpha^{\mathcal{H}_{E5}}_{f,\phi}$
$\beta^{\mathcal{H}_{E5}}_{E5}$
$\alpha^{\mathcal{H}_{E5}}_{f,E5^*}$
$D^{\mathcal{H}_{E5}}_{f,E5}$
$\mathcal{H}_{E5}$
$\alpha^{\mathcal{H}_{S5North}}_{f,E5^*}$
$\beta^{\mathcal{H}_{S5North}}_{S5North}$
$\alpha^{\mathcal{H}_{S5North}}_{f,S5North^*}$
$D^{\mathcal{H}_{S5North}}_{f,S5North}$
$\mathcal{H}_{S5North}$
$\alpha^{\mathcal{H}_{S4East}}_{f,S5North^*}$
$\beta^{\mathcal{H}_{S4East}}_{S4East}$
$\alpha^{\mathcal{H}_{S4East}}_{f,S4East^*}$
$D^{\mathcal{H}_{S4East}}_{f,S4East}$
$\mathcal{H}_{S4East}$
$\alpha^{\mathcal{H}_{S3South}}_{f,S4East^*}$
$\beta^{\mathcal{H}_{S3South}}_{S3South}$
$\alpha^{\mathcal{H}_{S3South}}_{f,S3South^*}$
$D^{\mathcal{H}_{S3South}}_{f,S3South}$
$\mathcal{H}_{S3South}$
Prop. 5.1
$\alpha^{\mathcal{H}_{S6East}}_{f,S3South^*}$
$\beta^{\mathcal{H}_{S6East}}_{S6East}$
$D^{\mathcal{H}_{S6East}}_{f,S6East}$
$\mathcal{H}_{S6East}$
$D^{\mathcal{H}_{TAI}}_{f,E5}$
$D^{\mathcal{H}_{TAI}}_{f,S5North}$
$D^{\mathcal{H}_{TAI}}_{f,S4East}$
$D^{\mathcal{H}_{TAI}}_{f,S3South}$
$D^{\mathcal{H}_{TAI}}_{f,S6East}$
$\mathcal{H}_{TAI}$
Legend:
Proposition 5.3
Proposition 5.1
Theorem 2.2

Figure 5.8: Illustration of the strategy "always in local time" for the example of Figure 5.6.

(sometimes in TAI, sometimes in the local clock).

5.4.2 Networks with Non-ideal Clocks and PREOFs

When a node contains an optional function such as a PEF, a POF, or a REG, the corresponding model that we have developed in Section 4.2.2 of Chapter 4 is *a priori* only valid when observed with the local clock of the output port. To apply the "always in TAI" strategy, we need to obtain a model for the same function but as observed with $\mathcal{H}_{\text{TAI}}$.

The PEFs: Consider for example a PEF for f at a node n that is modeled by $\texttt{PEF}^{\mathcal{H}_n}_n(f)$ defined in Section 4.2.2 when observed with $\mathcal{H}_n$. Do we know a model for the same PEF, but as observed with $\mathcal{H}_{\text{TAI}}$?

The answer is yes, because the PEF does not need to measure elapsed time, thus its behavior is identical in $\mathcal{H}_{\text{TAI}}$, *i.e.*, $\texttt{PEF}^{\mathcal{H}_{\text{TAI}}}_n(f) = \texttt{PEF}^{\mathcal{H}_n}_n(f)$ is also a model for the same PEF observed with $\mathcal{H}_{\text{TAI}}$. Hence Theorem 4.1 can be applied to the model $\texttt{PEF}^{\mathcal{H}_{\text{TAI}}}_n(f)$ in $\mathcal{H}_{\text{TAI}}$. Its application requires that the arrival curves and delay bounds of its formulation be provided as observed with $\mathcal{H}_{\text{TAI}}$. This requirement is trivial in the context of the "always in TAI" strategy because all the intermediate results are obtained in $\mathcal{H}_{\text{TAI}}$ (Figure 5.7).

The POFs: Consider now a POF for $\mathcal{F}$ at a node n with reference o that is modeled by $\texttt{POF}^{\mathcal{H}_n}_n(\mathcal{F}, o)$ with timeout $T^{\mathcal{H}_n}$ as defined in Section 4.2.2 when observed with $\mathcal{H}_n$. Do we know a model for the same POF, but as observed with $\mathcal{H}_{\text{TAI}}$?

The answer here is more complex. Indeed, the only model that we know from the results of the current chapter is the model $\texttt{POF}_n^{\mathcal{H}_{\text{TAI}}}(\mathcal{F}, o)$ with the same behavior as in Figure 4.8 but with a timeout $T^{\mathcal{H}_{\text{TAI}}}$ that can vary in $[T_{\text{min}}^{\mathcal{H}_{\text{TAI}}}, T_{\text{max}}^{\mathcal{H}_{\text{TAI}}}]$ where $T_{\text{min}}^{\mathcal{H}_{\text{TAI}}}$ and $T_{\text{max}}^{\mathcal{H}_{\text{TAI}}}$ are obtained from Proposition 5.1. As we do not know any result that applies expressly to a POF model with varying timeout[3], we cannot apply the “always in TAI” strategy. We would typically apply here the “local time” strategy to apply the results from [Mohammadpour, Le Boudec 2021] in the POF's local time.

The REGs: The next section is devoted to the analysis of REGs with non-ideal clocks.

5.5 Regulators in Networks With Non-Ideal Clocks

As we discussed in the introduction of the chapter, the internal operations of a regulator require to measure elapsed time. Therefore, like the POF, a system that is a regulator when observed with its own internal clock might no longer meet the definition of a regulator when observed with a different (e.g., external) clock.

Assume for instance that a PFR is configured with a shaping curve σ_f for flow f. This shaping curve is enforced with the regulator's internal clock $\mathcal{H}_{\texttt{REG}}$ and σ_f is an arrival curve for f at the output of the regulator, but when observed with the regulator's internal clock, *i.e.*, $\alpha_{f,\texttt{REG}*}^{\mathcal{H}_{\texttt{REG}}} = \sigma_f$. Further assume that the PFR's clock runs faster than expected when observed with an external clock. Then the external observer will see that the system's output violates the configured shaping curve thus the system cannot be a regulator. Conversely, if the clock of the system is too slow from an external point of view, then the external observer will conclude that the system is not a regulator because it does not release the packets as soon as possible. Worse, the clock may oscillate between being too fast or being too slow.

Similar observations can be done with the IR. Consequently, none of the properties of the regulators (Section 3.1.5 of Chapter 3) are expected to hold when observed with an external clock. In particular, the shaping-for-free property might be violated.

Using Proposition 5.3 and Table 5.1, one can obtain the arrival curve of the flow $\alpha_{f,\texttt{REG}*}^{\mathcal{H}_g}$ as observed with an external clock $\mathcal{H}_g$ based on the arrival curve $\alpha_{f,\texttt{REG}*}^{\mathcal{H}_{\texttt{REG}}} = \sigma_f$ as observed with $\mathcal{H}_{\texttt{REG}}$. In this section, we hence focus on the second issue, *i.e.*, the violation of the shaping-for-free property due to non-ideal clocks.

In the following, we say that a regulator is non-adapted if its configured shaping curve(s) do not take into account the clock non-idealities: the regulators are configured as if the clocks in the network were ideal. Proposition 5.5 proves that in non-synchronized networks, the shaping-for-free property holds neither for a non-adapted PFR nor for a non-adapted IR. For synchronized networks, Section 5.5.3 computes a lower and an upper bound on the worst-case penalty incurred by a non-adapted PFR. Finally, Proposition 5.10 proves that the IR cannot provide any delay bound even in networks with arbitrary tight synchronization.

[3]We could argue that some, if not all, of the results of [Mohammadpour, Le Boudec 2021] can be extended to a POF with a varying timeout but no formal extension has been performed.

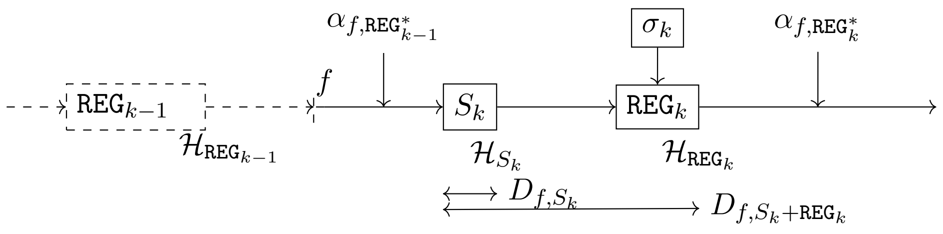

$\alpha_{f,\mathtt{REG}^*_{k-1}}$
σ_k
$\alpha_{f,\mathtt{REG}^*_k}$
$\mathtt{REG}_{k-1}$
f
S_k
$\mathtt{REG}_k$
$\mathcal{H}_{\mathtt{REG}_{k-1}}$
$\mathcal{H}_{S_k}$
$\mathcal{H}_{\mathtt{REG}_k}$
D_{f,S_k}
$D_{f,S_k+\mathtt{REG}_k}$

Figure 5.9: Network model for a regulated flow. At each hop, the flow goes through a system S_k and is then regulated by the regulator $\mathtt{REG}_k$. Each system S_k and reach regulator $\mathtt{REG}_k$ has its own internal clock, noted at the bottom right.

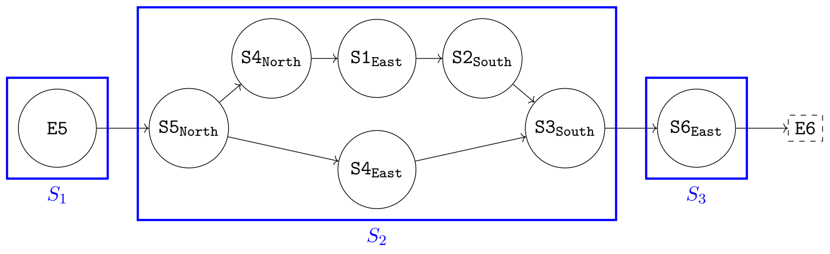


Figure 5.10: An example of mapping between the flow graph of the toy example used in Chapter 4 (Figure 4.6) and the sequence of S_k systems used in the current chapter.

5.5.1 Simplified Network Model For The Analysis of Regulators in Networks with Non-ideal Clocks

In the previous chapters, we have used the model of a TSN device described in Chapter 2, Section 2.6. The results that we introduce in this chapter are of interest beyond the scope TSN model of Chapter 2. Therefore, we present here as slightly different network model.

We assume that the path of a flow f is a sequence of systems S_k. Each S_k can contain any assembly of network elements other than the regulators for f but each S_k must remain causal and FIFO for f and its output must be packetized. The flow f is assumed to be processed by a regulator after each system in its path, except the last one (Figure 5.9). We call "kth hop" of this flow the sequence $(S_k - \mathrm{REG}_k)$. By convention, $\mathtt{REG}_0$ denotes the source of the flow, its output is the observation point noted as ϕ in the rest of the manuscript.

Remark (Relation with Chapter 4): A system S_k in the current chapter can be modeled by an assembly made of a packet-replication function, several parallel paths and a packet-elimination function of Chapter 4, as illustrated in Figure 5.10. Thus we can combine the results of Chapter 4 with the results of the current chapter. The assumption that S_k is causal and FIFO for f are respectively translated, in the context of Chapter 4, into (a) the first vertex within S_k must be a diamond ancestor of the last vertex within S_k, and the latter shall not be an EP-vertex, and (b) a POF for f is placed at the last vertex within S_k to correct for any mis-ordering caused by the redundancy.

Note that we assume (b) because when it does not hold, we have already discussed in Chapter 4 that regulators incur delay penalties.

For each flow f and each index $k = 1 \dots n$, we note $\mathtt{REG}_k$ the regulator (PFR or IR) that processes flow f after the network element S_k. In the previous chapters, a regulator enforce a shaping curve defined by a reference point: the shaping curve is the arrival curve that the flow has at the reference. In this chapter, we allow to freely select the rate and burst parameters of the regulators to take into account the clock inaccuracies. On $\texttt{REG}_k$, we assume that we can configure a shaping curve σ_k with a burst b_{Reg_k} and a rate r_{Reg_k} for this flow. Practical implementations support only limited accuracy, and we note $\mathcal{Q}_k(b)$ [resp. $\mathcal{R}_k(r)$] the lowest value that is configurable by $\texttt{REG}_k$ and that is higher than b [resp r].

For each system S, we assume that if we know the arrival curve of each flow g going through S, observed with $\mathcal{H}_{\mathrm{TAI}}$, then for any flow f going through S we can obtain a delay bound $D_{f,S}^{\mathcal{H}_{\mathrm{TAI}}}$ for the flow through the system S. This delay bound can be obtained using the network-calculus results (Chapter 2) applied in the $\mathcal{H}_{\mathrm{TAI}}$ clock. Each system and each regulator (including, for each flow f, its source $\texttt{REG}_0$) uses a local clock noted at the bottom right, as in Figure 5.9. Each flow f exits its source $\texttt{REG}_0$ with a leaky-bucket arrival curve of rate $r_0 = r_{\mathrm{Reg}_0}$ and burst $b_0 = b_{\mathrm{Reg}_0}$, when observed with $\mathcal{H}_{\mathrm{Reg}_0}$.

The delay bounds provided in this section are only valid for the stream of non-lost data units of f.

5.5.2 Non-synchronized Networks with Regulators

We now combine the set of results of Section 5.3 with other network-calculus results to analyze non-synchronized networks containing regulators. We focus on a network with flows that are leaky-bucket constrained at their sources.

Instability of Non-Adapted Regulators in Non-Synchronized Networks

Proposition 5.5 (Instability of non-adapted regulators in non-synchronized networks) *Consider a non-synchronized network with $\rho > 1$ and $\eta \geq 0$. Consider a flow f and system S_k as in Figure 5.9 that is causal, FIFO for f and that guarantees to f a delay upper-bounded by $D_{f,S_k}^{\mathcal{H}_{TAI}}$ if the flow satisfies an arrival curve $\alpha_{f,S_k^{in}}^{\mathcal{H}_{TAI}} = \gamma_{r,b}$ when observed with $\mathcal{H}_{TAI}$. The flow is submitted to a non-adapted regulator (PFR or IR) with shaping curve $\sigma_k \triangleq \gamma_{r,b}$ (Figure 5.9). There exist adversarial source clocks within our time model and adversarial traffic generation that satisfies the $\gamma_{r,b}$ arrival curve such that the delay of the flow through the regulator, measured using any clock of the network, is unbounded.*

Recall that if the clocks would be ideal, the worst-case delay of the flow would not be increased by the regulator (Chapter 3). In contrast, with nonideal clocks, the total delay is unbounded, thus the "shaping-for-free" property does not hold for non-adapted regulators.

The proof in Appendix B.3.5 is inspired by a remark made in the TSN ATS standard [IEEE 802.1Qcr, Annex V.8]: "*If the upstream device [...] runs faster than nominal and [the] downstream Bridge [...] runs slower than nominal, the backlog as well as the per hop delay in the downstream Bridge could grow under peak conditions*". In our model, the "upstream device" represents the upstream regulator, or the source, the "downstream Bridge" represents the next regulator, and "peak conditions" refer to a greedy source.

The Rate and Burst Cascade for Non-Synchronized Networks with Regulators

This last quoted remark highlights how a first solution for the instability problem can be formulated: one can make sure that whatever the clock conditions (but within the constraints of Equation (5.5)), the downstream device will always have an output rate higher than the input. This requires increasing slightly the nominal rate of the regulator; and because this increase is performed at every hop, it generates a rate and burst cascade that was first described in [IEEE 802.1Qcr, Annex V.8] and that we refine here, proving also its validity. The rate and burst cascade works as follows.

Step 1: For each flow f, and each hop $k = 1 \dots n$ in its path, configure $\texttt{REG}_k$ with $r_{\texttt{REG}_k} = \mathcal{R}_k(\rho r_{\texttt{REG}_{k-1}})$ and $b_{\texttt{REG}_k} = \mathcal{Q}_k(b_{\texttt{REG}_{k-1}} + \eta r_{\texttt{REG}_{k-1}})$. Recall that $\mathcal{R}_k(r)$ [resp. $\mathcal{Q}_k(b)$] denote the smallest configurable rate [resp. burst] higher than r [resp. b] for this regulator. Also, ρ and η are network-wide parameters. They depend neither on the considered clock nor on k.

Step 2: For each flow f, and each hop $k = 1 \dots n$ in its path, the configured shaping curve σ_{k-1} is an arrival curve at the output of the regulator $\texttt{REG}_{k-1}$, when observed with the clock of the regulator: $\alpha_{f,S_k^{\text{in}}}^{\mathcal{H}_{\texttt{REG}_{k-1}}} = \alpha_{f,\texttt{REG}_{k-1}^*}^{\mathcal{H}_{\texttt{REG}_{k-1}}} = \sigma_{k-1}$. Using Table 5.1 with $\mathcal{H}_g = \mathcal{H}_{\text{TAI}}$ and $\mathcal{H}_i = \mathcal{H}_{\texttt{REG}_{k-1}}$ shows that flow f has a leaky-bucket arrival curve $\alpha_{f,S_k^{\text{in}}}^{\mathcal{H}_{\text{TAI}}} = \gamma_{\rho r_{\texttt{REG}_{k-1}}, b_{\texttt{REG}_{k-1}} + \eta r_{\texttt{REG}_{k-1}}}$.

For each system S in the network and each flow f' crossing S, the arrival curve of f' at the input of S, observed with $\mathcal{H}_{\text{TAI}}$ is given by the above $\alpha_{f',S_k^{\text{in}}}^{\mathcal{H}_{\text{TAI}}}$, with k the index of S in the path of flow f'. According to the assumptions of Section 5.5.1, we can compute a TAI delay bound $D_{f,S}^{\mathcal{H}_{\text{TAI}}}$ of any flow f that goes through S.

Step 3: For each flow f, and each hop $k = 1 \dots n$ in its path, its TAI delay bound $D_{f,S_k+\texttt{REG}_k}^{\mathcal{H}_{\text{TAI}}}$ through the sequence $(S_k - \texttt{REG}_k)$ is given by the next proposition:

Proposition 5.6 (Hop delay with the rate-and burst cascade)
If the regulators are configured as in Step 1, for each flow f that goes through system S_k, a bound on the TAI delay of the flow through the concatenation of S_k and the next regulator is $D_{f,S_k+\texttt{REG}_k}^{\mathcal{H}_{TAI}} = \rho^2 D_{f,S_k}^{\mathcal{H}_{TAI}} + \eta(1+\rho)$ where $D_{f,S_k}^{\mathcal{H}_{TAI}}$ is a bound on the TAI delay of flow f through S_k, computed in Step 2.

The proof is in Appendix B.3.6. Intuitively, with the clock of the regulator, the arrival curve of the flow at the input of S_k is dominated by the shaping curve of the regulator, due to the inflation of rate and burst. Hence the shaping-for-free property holds in $\mathcal{H}_{\texttt{REG}}$ for the rate-and-burst-adapted regulator.

The rate and burst cascade has the drawback that the configuration of a regulator depends on its position in the flow path. This puts complexity on the control plane, specifically, for computing, distributing and managing the regulators' configuration.

Furthermore, it leads to a pessimistic reservation of rates. Indeed, every regulator $\texttt{REG}_k$ is conservatively configured so as to be stable even if the previous regulator $\texttt{REG}_{k-1}$ is greedy (its output matches exactly the shaping curve) and even if the clock of $\texttt{REG}_{k-1}$ is ρ times faster than the clock of $\texttt{REG}_k$, the maximum frequency difference allowed by Equation (5.5). But $\texttt{REG}_{k-1}$ is again configured in case $\texttt{REG}_{k-2}$ is greedy and with a clock ρ times faster than $\texttt{REG}_{k-1}$. All these worst-case situations cannot remain simultaneously for a long duration: Equation (5.5) applies for any pair of clock in the network and in particular it forbids $\mathcal{H}_{\texttt{REG}_{k-2}}$

to sustain a frequency ρ^2 times faster than $\mathcal{H}_{\texttt{REG}_k}$, even if the configuration of the rate-and-burst cascade would apply in this situation.

Theses remarks motivates an alternative method which, however, works only with PFRs.

The ADAM method for Non-Synchronized Networks with Per-Flow Regulators

The goal of the asynchronous dual arrival-curve method (ADAM) is, for any given flow, to have the same parameters at all regulators along the flow path. Hence we require that the rounding functions $\mathcal{R}_k, \mathcal{Q}_k$ (which capture the accuracy with which regulator parameters are actually implemented) are the same at all network nodes (and we consequently drop the index k for these functions). We also require that all the regulators in the network be PFRs.

The main idea of ADAM is to establish that each flow f has an arrival curve, expressed in $\mathcal{H}_{\text{TAI}}$, of the form $\alpha^{\mathcal{H}_{\text{TAI}}}_{f,\texttt{REG}^*_k} = \alpha_{f,1} \otimes \alpha_{f,2,\texttt{REG}^*_k}$ at the output of its k-th hop where $\alpha_{f,1}$, $\alpha_{f,2,\texttt{REG}^*_k}$ are leaky-bucket arrival curves and the former is independent of the hop index.

Step 1 (per-flow): For each flow f, find a rate margin W such that $W \geq \rho^2$ and $\mathcal{R}(Wr_0) = Wr_0$. Configure the shaping curves at all regulators along the path of the flow with rate $r_{\texttt{REG}_k} = Wr_0$ and burst $b_{\texttt{REG}_k} = b_0$. Since ρ is a network-wide parameter that does not depend on a clock, all regulators except the source have the same configuration, independent of the hop index $k = 1...n$. Here r_0, b_0 are the rate and burst at the source (which depend on f, though the dependency on f is not shown for simplicity of notation).

Step 2 (per-node): Any flow f that goes through a system S is output by the regulator at its previous hop. Thus, using the same justification as in Step 2 of the rate and burst cascade, it has the arrival curve $\alpha^{\mathcal{H}_{\text{TAI}}}_{f,1} = \gamma_{\rho W r_0, b_0 + \eta W r_0}$ (i.e. leaky bucket with rate $\rho W r_0$ and burst $b_0 + \eta W r_0$), when observed with $\mathcal{H}_{\text{TAI}}$.

Then, again using the same justification as in Step 2 of the rate and burst cascade, compute a TAI delay bound $D^{\mathcal{H}_{\text{TAI}}}_{f,S}$ through any system S and for any flow f that goes through it.

Step 3 (per-flow): For each flow f and each hop $k = 1 \dots n$ in its path, compute a TAI delay bound, $D^{\mathcal{H}_{\text{TAI}}}_{f,S_k + \texttt{REG}_k}$, of the flow through the sequence $S_k - \texttt{REG}_k$), using Algorithm 2.

Proposition 5.7 (Correctness of Algorithm 2)
For a flow f that has n hops and for $m = 1 \dots n$:

1. *Let $\alpha_{f,2,REG^*_0}$ be the leaky-bucket curve with rate r_2 (Line 2) and burst b_{2,REG^*_0} (Line 3); it is an arrival curve for the flow at its source when observed with $\mathcal{H}_{TAI}$.*
2. *Let α_{2,REG^*_k} be the leaky-bucket curve with rate r_2 (Line 2) and burst b_{2,REG^*_k} (Line 6); it is an arrival curve for f observed with $\mathcal{H}_{TAI}$ at the output of REG_k.*
3. *$D^{\mathcal{H}_{TAI}}_{f,S_k+REG_k}$ (Line 5) is a TAI delay bound for the flow through the concatenation $(S_k - REG_k)$, for $k = 1 \dots m$.*

The proof is in Appendix B.3.7. Unlike with the rate and burst cascade method, here the regulators increase slightly the delay bound, specifically, the shaping-for-free property does not hold. The proof captures this increase by using the service-curve characterization of the PFRs (Proposition 3.1), together with the arrival curve $\alpha_{f,1} \wedge \alpha_{f,2,\texttt{REG}^*_{k-1}}$ for the flow at the input of S_k observed with $\mathcal{H}_{\text{TAI}}$. It then applies Theorem 2.2 in $\mathcal{H}_{\text{TAI}}$. As we do not know any service-curve characterization for IRs, we are not able to extend this method to IRs.

Algorithm 2 Computing the TAI delay for a flow through its hop m using the ADAM method

```
Require: $\{D^{\mathcal{H}_{\mathrm{TAI}}}_{f,S_k}\}_k$, the set of all the TAI delay bounds for the flow through the systems $\mathrm{S}_k$ in its path (from Step 2).
Require: $m$, the index at which to compute the TAI hop delay.
Require: $r_0$, $b_0$, the rate and burst of the flow at the source, observed with the source's clock. $W$, the rate margin selected at Step 1.
Require: $\eta$, $\rho$ the network-wide parameters of the time-model.
1: function ComputeDelayBoundForHopM($\{D^{\mathcal{H}_{\mathrm{TAI}}}_{f,S_k}\}_k$,$m$,$(r_0, b_0)$,$W$,$\rho$,$\eta$)
2:     $r_2 \leftarrow \rho r_0$
3:     $b_{2,\mathtt{REG}^*_0} \leftarrow b_0 + \eta r_0$
4:     for $k = 1 \dots m$ do
5:         $D^{\mathcal{H}_{\mathrm{TAI}}}_{f,S_k+\mathtt{REG}_k} \leftarrow D^{\mathcal{H}_{\mathrm{TAI}}}_{f,S_k} + \eta(1+\rho) + \frac{b_{2,\mathtt{REF}^*_{k-1}} - b_0 - \eta W r_0}{\rho r_0} \frac{\rho^2-1}{W-1}$
6:         $b_{2,\mathtt{REG}^*_k} \leftarrow b_{2,\mathtt{REG}^*_{k-1}} + r_2 \cdot D^{\mathcal{H}_{\mathrm{TAI}}}_{f,S_k+\mathtt{REG}_k}$          ▷ See Proposition 5.7
7:     end for
8:     return $D^{\mathcal{H}_{\mathrm{TAI}}}_{S_m+\mathtt{REG}_m}$
9: end function
```

Also note that the delay bound in Step 2 is computed using the arrival curve $\alpha_{f,1}$ and not the full arrival curve known by the method. This is because Step 2 is performed at every node in the network, before knowing the results of Step 3 that is performed per-flow. Using the result of Step 3 in Step 2 is possible in feed-forward networks and might in some cases lead to slightly smaller delay bounds. However, one of the major applications of regulators is in non-feedforward networks. Therefore, we do not explore such possible optimizations.

5.5.3 Synchronized Networks with Regulators

In synchronized networks, we expect that unbounded delays due to non-adapted regulators cannot occur, as clock rates cannot diverge for long periods of time. We now examine to which extent this holds. We study separately PFRs and IRs.

Delay Penalty of Non-Adapted PFRs in Synchronized Networks

Even when synchronized, a non-adapted PFR increases the worst-case delay

Proposition 5.8 (Lower bound on the worst-case delay penalty of synchronized non-adapted per-flow regulators)
For any leaky-bucket arrival curve $\gamma_{r,b}$, there exists a network configuration such that 1/ one flow has the arrival curve $\gamma_{r,b}$ at its source, when observed with the source clock, 2/ the flow goes through one network element followed by one non-adapted PFR (hence with shaping curve $\gamma_{r,b}$) 3/ the clocks of the source and the PFR are synchronized with time-error bound Δ, and 4/ the PFR increases the worst-case delay by at least Δ.

The proof is in Appendix B.3.8. Thanks to the synchronization, though, the delay penalty of the PFR can be controlled:

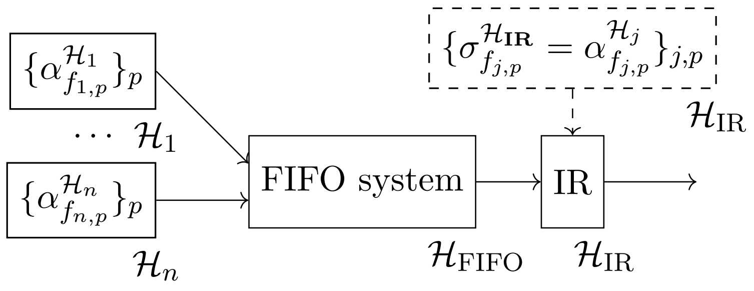

$\{\alpha^{\mathcal{H}_1}_{f_{1,p}}\}_p$
$\cdots$ $\mathcal{H}_1$
$\{\alpha^{\mathcal{H}_n}_{f_{n,p}}\}_p$
$\mathcal{H}_n$
FIFO system
$\mathcal{H}_{\mathrm{FIFO}}$
$\{\sigma^{\mathcal{H}_{\mathrm{IR}}}_{f_{j,p}} = \alpha^{\mathcal{H}_j}_{f_{j,p}}\}_{j,p}$
$\mathcal{H}_{\mathrm{IR}}$
IR
$\mathcal{H}_{\mathrm{IR}}$

Figure 5.11: Notations of Proposition 5.10. A non-adapted IR is placed after a causal, lossless, FIFO system that is connected to n sources, each with a different clock.

Proposition 5.9 (Upper bound on the delay penalty of synchronized non-adapted per-flow regulators)
For a synchronized network and a flow f, if $\texttt{REG}_k$ is a non-adapted PFR, then a TAI delay bound for f through the concatenation of S_k and $\texttt{REG}_k$ is $D^{\mathcal{H}_{TAI}}_{f,S_k+\texttt{REG}_k} = D^{\mathcal{H}_{TAI}}_{f,S_k} + 4\Delta$.

The proof is in Appendix B.3.9. Note that a TAI delay bound $D^{\mathcal{H}_{\mathrm{TAI}}}_{f,S_k}$ can be obtained by using the fact that $\alpha^{\mathcal{H}_{\texttt{REG}_{k-1}}}_{f,\texttt{REG}^*_{k-1}} = \sigma_{\texttt{REG}_{k-1}}$. Hence we can use the last line of Table 5.1. If $\texttt{REG}_{k-1}$ is also non-adapted (*i.e.*, enforces γ_{r_0,b_0}), then the flow enters S_k with a double arrival-curve constraint when observed with $\mathcal{H}_{\mathrm{TAI}}$: a leaky-bucket arrival curve of rate ρr_0 and burst $b_0 + r_0\eta$, and a leaky-bucket arrival curve of rate r_0 and burst $b_0 + 2r_0\Delta$.

Propositions 5.8 and 5.9 show that the worst-case penalty on the TAI per-hop delay of non-adapted PFRs in synchronized networks is between Δ and 4Δ, *i.e.*, it is of the order of magnitude of the synchronization precision. For tightly synchronized networks and PFRs, the current practice of ignoring clock non-idealities is thus perfectly acceptable. However, in loosely synchronized networks, the value of Δ ($\sim$ 125ms) is larger than the required delay bound for flows with stringent delay requirements. The two solutions (rate and burst cascade, and ADAM) that apply to non-synchronized networks also apply here and can be used.

Instability of Non-Adapted IRs in Synchronized Networks

For the interleaved regulator, however, the conclusions are very different. Indeed, the IR may yield unbounded latencies, even with tightly synchronized networks, as shown in the following proposition.

Proposition 5.10 (Instability of non-adapted synchronized interleaved regulators)
Consider an interleaved regulator as in Figure 5.11, with n upstream systems. Assume that (a) each upstream system j outputs P_j flows $\{f_{j,p}\}_{p=1}^{P_j}$, each with a known arrival curve $\alpha^{\mathcal{H}_j}_{f_{j,p}}$ when observed with clock $\mathcal{H}_j$; (b) the interleaved regulator is non-adapted: $\forall(j,p), \sigma^{\mathcal{H}_{IR}}_{f_{j,p}} = \alpha^{\mathcal{H}_j}_{f_{j,p}}$; and (c) the clocks $\mathcal{H}_{IR}$,$\mathcal{H}_{FIFO}$ and each of the $\mathcal{H}_j$ are synchronized with the synchronized model with parameters η, ρ, Δ (as per Equations (5.5) and (5.6)). Then, for any parameters n, ρ, η, Δ with $n \geq 3$, $\rho > 1$, $\eta \geq 0$ and $\Delta > 0$, there exists a FIFO system, adversarial clocks for $\mathcal{H}_{IR}$,$\mathcal{H}_{FIFO}$ and $\{\mathcal{H}_j\}_j$, and adversarial traffic generation of the upstream systems, such that the flows crossing the IR have unbounded latency within the IR, when observed with any clock of the network.

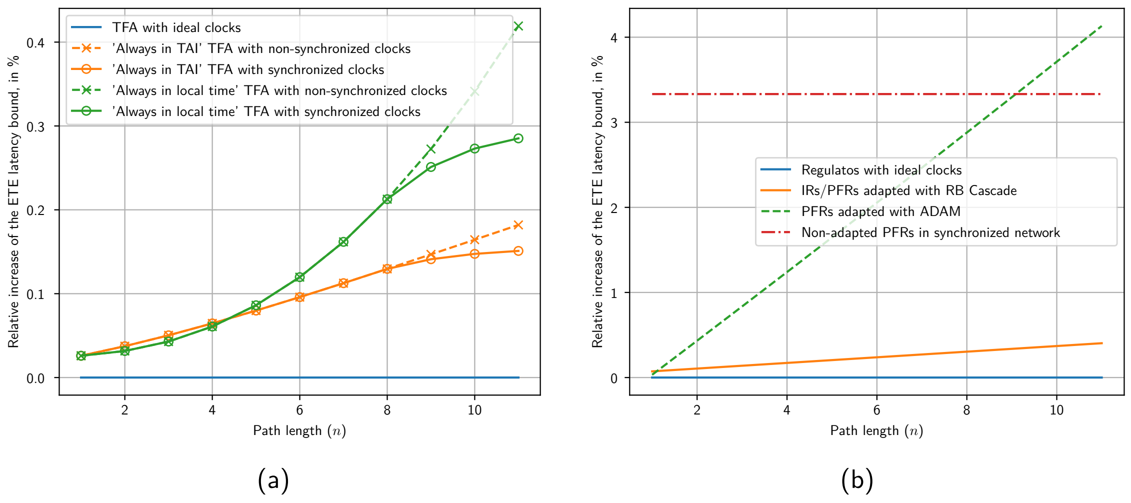


Figure 5.12: Comparison of the end-to-end strategies and of the adaptation methods. (a) End-to-end latency bounds as a function of the path length, obtained either with the "always in TAI" strategy or with the "always in local time strategy", in synchronized and non-synchronized networks. (b) End-to-end latency bounds obtained with different adaptation methods for the regulators, when each node in the flow path is followed by a regulator.

The proof is in Appendix B.3.10.

Numerical application to IEEE TSN: In the proof, the delay divergence increases at a rate $\sqrt{\rho} - 1$ for any number of previous sources $n \geq 3$, and any synchronization precision $\Delta > 0$. This corresponds to 100μs of increased worst-case delay per second of network operation for both tightly- or loosely synchronized networks.

5.6 Evaluation of the Approaches

In this section, we evaluate the two proposed strategies as well as the proposed regulator adaptation methods on a parametric topology.

We consider a unique flow f that crosses n systems $S_1, \dots, S_n$. Because the model developed in this chapter is independent from the model of a TSN bridge, we simply assume here that the systems $(S_i)_i$ are service-curve elements. The flow's source and each of the $(S_i)_i$ has its own clock. When observed with its own clock, S_i offers to f a rate-latency service curve with rate 100Mbit/s and latency 1μs. When observed with its own clock, the source generates flow f with a leaky-bucket arrival curve with rate 80Mbit/s and burst of 1500Bytes.

5.6.1 Comparison of the Two End-to-end Strategies

We first use a classic TFA approach with no line shaping and no regulator, and we compute the end-to-end latency bounds obtained with TFA when the clocks are ideal, with a path length n between 1 and 11. The latency bounds for these ideal-clock situation range from 121μs to 97ms and are used as a reference (line at y=0 in Figure 5.12a).

Then we remove the ideal-clock assumption and compute the relative increase, with respect to the ideal-clock situation, of the latency bounds obtained either with the "always in TAI" or the "always in local time" strategies, with the non-synchronized ($\eta =$4ns, $\rho =$1.0002) and synchronized ($\Delta =$1μs) time models.

For a path of a short length, the "always in local time" strategy performs slightly better than the "always in TAI" strategy, but as the length of the path increase, the gain of the latter becomes significant with respect to the former.

As illustrated in Figure 5.8, the "always in local time" strategy often calls Proposition 5.3 that translates, in a non-synchronized network, a $\gamma_{r,b}$ arrival curve in one clock into a $\gamma_{\rho r, b+\rho\eta}$ in another clock. The burst increase due to this operation is small ($\rho\eta$) and the rate increase (ρr) does not immediately worsen the delay bound in the current node[4] but worsens the burstiness increase in the subsequent nodes. On the contrary, the "always in TAI strategy" (Figure 5.7) often calls Proposition 5.4 that translates, in a non-synchronized network, a $\beta_{R,T}$ service curve observed with a local clock into a $\beta_{R/\rho, \rho T+\eta}$. Both the decrease of the rate and the increase of the latency worsen the delay bound for the current node, especially as R is high.

This is why, when the path of the flow is short, the "always in local time" strategy performs better than the other one. However, as the length of the path increases, the cascade of worsened rates with the local-time strategy worsen the bursts and the delay penalty with respect to the other strategy becomes visible. Furthermore the delay bounds per node also increase which further increases the penalty that the local-time strategy suffers by also calling Proposition 5.1 to translate the delay bounds.

As the length of the path increases (and the delays become larger), the gain of the synchronization becomes visible. Overall, for reasonable path lengths, the effect of the clock non-idealities is negligible when no regulator is used (less than 0.4%) and the current practice of ignoring the non-idealities is acceptable.

5.6.2 Comparison of the Regulator Adaptation Methods

We now place a regulator (a PFR, since f is the only flow) after each system S_i in the flow path, as in Figure 5.9. We first compute the end-to-end latency bound obtained with the regulators when the clocks are ideal: the latency is simply n times 121μs with n the length of the path, because the regulators block the burstiness propagation. These latency bounds are used as a reference (line at $y = 0$ in Figure 5.12b).

We then remove the ideal-clock assumption and consider the non-synchronized time model. As discussed in the previous section, the parameters of the regulators must be adapted or it can incur unbounded latencies. We compare the end-to-end latency bounds obtained with the rate-and-burst cascade and with the ADAM method. The rate-an-burst cascade performs better than ADAM but both remain within a few percent of the ideal-clock situation.

Last we consider the synchronized time-model. In this case, the PFR can either be adapted with the rate-and-burst cascade or ADAM (both are not influenced by the presence

[4]Within a common clock, the network-calculus delay bound for a $\gamma_{r',b'}$-constrained flow inside a $\beta_{R',T'}$ service-curve element is $T' + b'/R'$, the rate r' of the leaky-bucket arrival curve does not influence the delay bound for this system.

of the synchronization), or it can be non-adapted, in which case it provides the worst latency bounds, but within a few percent of the ideal-clock situation.

Conclusion

In this chapter, we developed a framework for modeling non-ideal clocks in network calculus. The framework includes a time-model for the non-synchronized and synchronized networks (Section 5.2.1) and a toolbox of results for capturing the effects of the clock non-idealities in network calculus.

We have proposed two strategies ("always in TAI" and "always in local time", Section 5.4) for computing end-to-end latency bounds in networks with non-ideal clocks.

In loosely synchronized and in non-synchronized networks, regulators are affected by clock issues; this leads to large or unbounded delays if not properly addressed (Section 5.5). We have proposed two adaptation methods for per-flow regulators: the rate-and-burst cascade, and ADAM. The former imposes that the regulator parameters depend on the position of the regulator on the flow path, which complexifies the control plane. For interleaved regulators, only the rate and burst cascade applies.

In tightly synchronized networks, non-adapted per-flow regulators are affected, but the delay penalty is of the order of the synchronization precision and can be neglected. In contrast, interleaved regulators are affected well beyond the synchronization precision, and this can lead to unbounded delays if the issue is not properly addressed. Therefore, a solution such as the method of rate-and-burst cascade should be applied with interleaved regulators, even in tightly synchronized networks.

Our evaluation of the two different end-to-end strategies on a parametric topology (Section 5.6) confirms that the "always in TAI" strategy performs better than the other strategy, except for short paths. The time-synchronization mechanisms improves the TAI latency bounds with respect to non-synchronized networks when the paths of the flows are long.

Finally, we have observed that the rate-and-burst cascade is the adaptation method that provide the best latency bounds, less than 0.5% worse than the latency bounds obtained with ideal clocks. The use of non-adapted PFRs in a tightly synchronized network and the use of PFRs adapted with ADAM provide similar performance bounds. These bounds remain within a few percent of the ideal-clock situation.

This chapter concludes the theoretical contributions of this thesis. The second part of this thesis will focus on the practical tools. Chapter 6 presents experimental modular TFA (xTFA), a tool that implements the theoretical contributions of this thesis and can compute end-to-end latency bounds in networks with cyclic dependencies, traffic regulators, PREFs and non-ideal clocks. Chapter 7 proposes a modification to the ns-3 network simulator that is used to validate the instability of the IR in a synchronized network with non-ideal clocks (Proposition 5.10 of the current chapter). Finally in Chapter 8 we validate the applicability of our results and the flexibility of the xTFA tool through an industrial use-case.

Part III

Practical Contributions

Chapter 6

xTFA: A Tool for Computing Delay Bounds in Time-Sensitive Networks

I don't think necessity is the mother of invention — invention, in my opinion, arises directly from idleness, possibly also from laziness.

Agatha Christie, *An Autobiography.*

Contents

6.1 Overview of xTFA 119

6.1.1 General Requirements of xTFA 120

6.1.2 Main Features of xTFA 120

6.2 Main Original Ideas in xTFA for Feed-Forward Networks 121

6.2.1 Breaking the TFA Approach into a Three-Step Process 121

6.2.2 The Concept of Flow State 123

6.2.3 The Computation Pipelines 126

6.2.4 Modeling the Line Shaping and the Packet Elimination Function Using aggregate computation pipeline (ACP) Computation Blocks 129

6.3 Adaptations of xTFA for Cyclic Dependencies 134

6.3.1 Restrictions on xTFA for Networks with Cyclic Dependencies 135

6.3.2 Latency Bounds in Networks With Cyclic Dependencies 135

In this chapter we present *experimental modular TFA* (xTFA), a tool for computing latency bounds in time-sensitive networks. As the name suggests, xTFA is an experimental tool that has been developed throughout the thesis for assessing the effect of cyclic dependencies, redundancy mechanisms and clock non-idealities on latency-bounds, based on the theoretical results presented in the previous chapters.

On one hand, xTFA designates a set of original algorithms and data-structures that are presented in this chapter. On the other hand, it also designates an implementation of these algorithms using Python [20].

In this chapter, we first introduce xTFA through its requirements and key features in Section 6.1. We then present the main xTFA data structures and algorithms in Section 6.2. In Section 6.3 we finally discuss the modifications made on xTFA when computing latency bounds in networks with cyclic dependencies.

6.1 Overview of xTFA

This section presents an overview of xTFA through its requirements and key features.

6.1.1 General Requirements of xTFA

The main goal of xTFA is to provide a tool for applying the theoretical results obtained throughout the thesis on realistic industrial networks. The main requirement of xTFA is hence:

1. xTFA shall use the theoretical contributions of the current thesis and shall compute latency bounds in networks that contain PREFs, POFs, regulators, non-ideal clocks and/or cyclic dependencies.

But xTFA has also been developed throughout the thesis, in parallel with the theoretical results. It has been used to gain insights on the behavior of the mechanisms, with a mutual influence between theory and practical implementation. Thus xTFA is also required to provide a very small "code-to-plots" time, *i.e.*,

2. The duration elapsed between the start of the implementation of a new result among the thesis's theoretical contributions and the obtention of graphical, even partial, results on use-cases shall be as low as possible.

For our paper [Thomas, Le Boudec, Mifdaoui 2019], we developed a tool that implements the FP-TFA algorithm described in Chapter 3. This tool was developed for obtaining delay bounds in networks that can only contain cyclic dependencies and/or regulators. It provides quick results but is not flexible. In particular, modeling PREOFs mechanisms (Chapter 4) requires to model nodes that might contain regulators, or POFs, or PEFs, or a combination of these "optional functions". This was not anticipated, and the tool based on FP-TFA that did not have the necessary flexibility for modeling them. This experience provides the two additional requirements for xTFA:

3. The tool shall be modular with respect to the optional functions: adding a model for a new optional function should be possible with minimal work.
4. It shall be possible to model a node that contains a combination of optional functions.

Note that xTFA is only an experimental tool used to support the theoretical contributions of the thesis. Thus, the time complexity, the memory complexity and the tightness of the results were never explicit requirements and were treated with a "best-effort" approach.

6.1.2 Main Features of xTFA

xTFA is based on the total-flow analysis (TFA) approach (Chapter 2, Section 2.5.1). It includes the improvements that were already included in FP-TFA (Chapter 3, Section 3.3), *i.e.*, the line-shaping effect [Grieu 2004; Mifdaoui, Leydier 2017], the service-curve of the packetizer (Theorem 3.1) to compute its effect on burst bounds and the improvement from [Mohammadpour, Stai, Le Boudec 2019]. It also includes the majority of the theoretical contributions of this thesis.

xTFA focuses on one class of a FIFO-per-class network. The network can be feed-forward or non-feed-forward. Each flow is assumed to be constrained by a known arrival curve at its source, when observed with the source's clock. Each CBQS is assumed to provide a known left-over service curve for the class of interest[1] when observed with the output-port's clock. The current version supports leaky-bucket-constrained flows and rate-latency-constrained CBQSs, but adding other curves for feed-forward networks only requires extending the min-plus back-end.

The clocks of the network are either assumed to be ideal, non-synchronized or synchronized as per the time model of Chapter 5 with known values for the time-jitter bound (η), clock-stability bound (ρ) and synchronization precision (Δ), if applicable. At the time of this writing, xTFA uses the "always in TAI" strategy (Chapter 5, Section 5.4.1) but the other strategy can also be implemented.

The network can contain packet-replication functions (PRFs) and packet-elimination functions (PEFs) as modeled in Chapter 4. It can also contain packet-ordering function (POF) as modeled in Chapter 4. The current version of xTFA assumes that the POF timeout is never fired, *i.e.*, the sequence of data units that reaches a POF is not incomplete[2].

The network can also contain traffic regulators (either PFRs or IRs) as modeled in Chapter 4, Section 4.2.2. Their parameters can also be adapted as per the rate-and-burst cascade or as per the ADAM method (Chapter 5, Section 5.5.2).

6.2 Main Original Ideas in xTFA for Feed-Forward Networks

To compute delay bounds in networks that can contain non-ideal clocks and output ports with optional functions, several original concepts, data structures and algorithms have been invented during the thesis and are presented in this section. A toy example is used to highlight the concepts:

Example: We consider the flow f of the toy example of Chapter 4 (last row of Table 4.2). Its graph is given in Figure 4.3: the flow is produced by the end system `E5`, is replicated by `S4` and sent over two paths. At `S3`, the duplicates of f are eliminated and the reconstructed flow reaches its destination `E6` through `S6`. We consider another flow g and we assume that g has the same path. Thus the output port `S3`$_{\texttt{South}}$ contains two packet-elimination functions (PEFs): one for f and one for g, as shown in Figure 6.1.

6.2.1 Breaking the TFA Approach into a Three-Step Process

Denote by $\mathcal{G}$ the graph induced by flows (GIF) of the network for the class of interest. If $\mathcal{G}$ is acyclic, the total-flow analysis (TFA) approach [Schmitt, Zdarsky 2006] iteratively computes delay bounds for the class-of-interest in every vertex of $\mathcal{G}$.

In this chapter, we say that a vertex n *can be computed* as per the TFA approach if every parent p of n in $\mathcal{G}$ *has been computed* (or if n has no parent in $\mathcal{G}$). And n *has been computed* if lower and upper delay bounds for the class of interest through n have been obtained and if an arrival curve for each individual flow $f \ni n$ has been obtained at n' and n^*: the output of the CBQS and the output of the packetizer on the remote input port, respectively.

[1]This left-over service curve can be computed using the related-work presented in Chapter 2, Section 2.4.1.

[2]This also removes the issues associated with the clock non-idealities of a POF that we have mentioned in Chapter 5, Section 5.4.2.

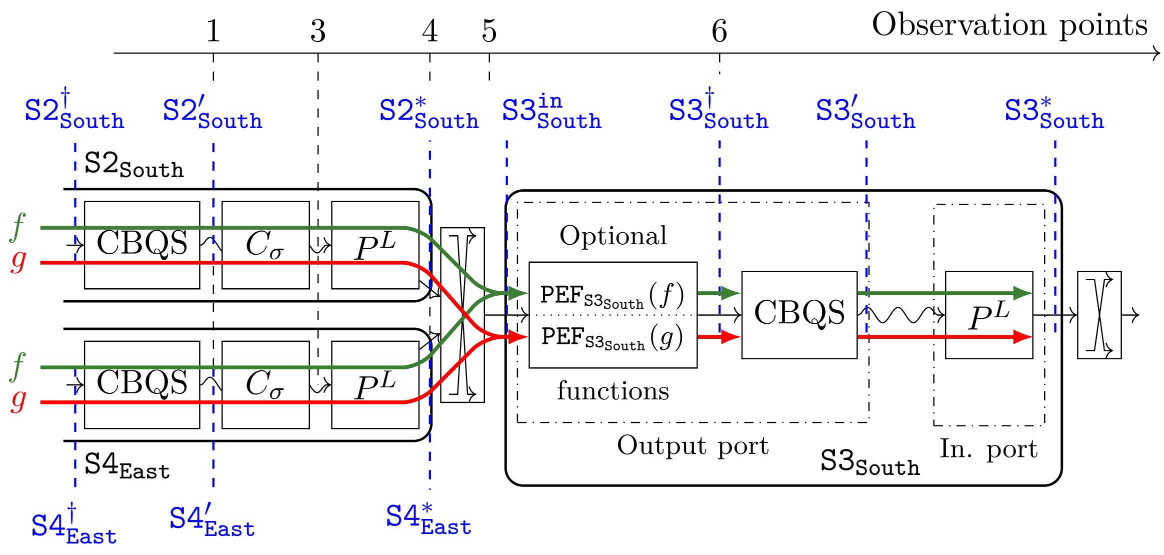


Figure 6.1: Overview of the TFA model for vertex $S3_{South}$ of the toy example in the last row of Table 4.2. The numbers on the top define observations points that are used later in the chapter.

For a vertex n that *can be computed*, the TFA approach proceeds in three steps:

• Step 1: Use the individual arrival curves $\{\alpha_{h,p^*}\}_{h,p}$ to compute an arrival curve $\alpha_{n^\dagger}$ for the aggregate of the class of interest at the input of the CBQS within n.

• Step 2: Use the aggregate arrival curve $\alpha_{n^\dagger}$ and the service curve β_n of the CBQS to compute a lower [resp., an upper] delay bound d_n [resp., D_n] for the aggregate through the CBQS. As the CBQS is assumed to be causal, lossless and FIFO, the delay bound for the aggregate is also a delay bound for each individual flow (Section 2.4.2).

• Step 3: Compute an arrival curve $\alpha_{h,n'}$ and α_{h,n^*} for individual flow $h \ni n$ at n' and n^*.

We refer to these steps as the *three-step* process, performed for every vertex n. In xTFA, the three steps are independent: The computations performed for one step can be changed without modifying the other two steps as long as each step follows the above specifications.

Example: Figure 6.1 shows the vertex $S3_{South}$ and its two parents $S2_{South}$ and $S4_{East}$, together with their respective CBQSs and with the path taken by f and g. Vertex $S3_{South}$ *can be computed* if and only if its two parents $S2_{South}$ and $S4_{East}$ *have been computed*. In that case, the individual arrival curves for f and g are known at $S2'_{South}$ and $S4'_{East}$: $\alpha_{f,S2'_{South}}$, $\alpha_{g,S2'_{South}}$, $\alpha_{f,S4'_{East}}$ and $\alpha_{g,S4'_{East}}$.
– Step 1 uses these four arrival curves to obtain an arrival curve for the aggregate made of f and g, noted $\alpha_{S3^{\dagger}_{South}}$, at $S3^{\dagger}_{South}$. To do so, it models all the mechanisms between $(S2'_{South}, S4'_{East})$ and $S3^{\dagger}_{South}$.
– Step 2 then computes delay bounds within the CBQS of vertex $S3_{South}$.
– Step 3 computes individual arrival curves $\alpha_{f,S3'_{South}}$, $\alpha_{g,S3'_{South}}$ at $S3'_{South}$ and $\alpha_{f,S3^{*}_{South}}$, $\alpha_{g,S3^{*}_{South}}$ at $S3^{*}_{South}$.

6.2.2 The Concept of Flow State

The first major modification in xTFA is the extension of the concept of individual arrival curves to the concept of *flow states*.

Presentation of the Concept

In FP-TFA (Section 3.3 in Chapter 3), Step 1 solely relies on the individual arrival curves of the flows to compute the aggregate arrival curve $\alpha_{n^\dagger}$. However, to compute the effect of a PEF on the aggregate arrival curve, we note that item 2/ of Theorem 4.1 in Chapter 4 requires the identification of the diamond ancestors. Furthermore, for each identified diamond ancestor a, Equation (4.1) requires the knowledge of a lower and an upper bound on the delay of the processed flow between a and n. Individual arrival curves are not sufficient to obtain this information. To cope with this issue, we extend any individual arrival curve with additional information to create the concept of a *flow state*.

Definition 6.1 (FlowState) *For a flow f, an observation point v, and a clock $\mathcal{H}$, the flow state $z^{\mathcal{H}}_{f,v}$ is a data structure that merges information on the* state *of flow f at the observation point v when observed with clock $\mathcal{H}$. The flow state is a collection:*

$$z^{\mathcal{H}}_{f,v} \triangleq (\alpha, \mathcal{U}, \{D[u]\}_{u\in\mathcal{U}}, \{d[u]\}_{u\in\mathcal{U}}, \{\lambda[u]\}_{u\in\mathcal{U}}) \tag{6.1}$$

that contains
- *$\alpha \triangleq \alpha^{\mathcal{H}}_{f,v}$, a leaky-bucket arrival curve of the flow f at the observation point v, when observed with clock $\mathcal{H}$*
- *$\mathcal{U}$ is a set of upstream vertices u for which the delay bounds of the data units of the f from the output u^* to the current observation point v is worth remembering. This set is used to identify the diamond ancestors. Formally, any vertex u of $\mathcal{G}(f)$ that is not an EP-vertex of $\mathcal{G}(f)$ and is located upstream of the observation point v can be a member of $\mathcal{U}$. Additionally, $\mathcal{U}$ contains the special object ϕ that represents the source application of the flow inside its source vertex. The members of this set are called* tags.
- *For each tag $u \in \mathcal{U}$, $d[u]$ [resp., $D[u]$] is a lower-bound [resp. an upper-bound] of the delay of the data units from u^* to v, along any possible paths $u \to v$ in $\mathcal{G}(f)$, when measured with clock $\mathcal{H}$.*
- *For each tag $u \in \mathcal{U}$, $\lambda[u] \triangleq \lambda^{\mathcal{H}}_{f,v}(u^*)$ is an upper-bound on the reordering late time offset (RTO) of the flow f at the observation point v, with respect to the order that the data units had at u^* and as observed by clock $\mathcal{H}$.*

Example: We consider the delay bounds of the toy example shown on Figure 6.2 with arbitrary time units measured with the TAI. We assume that there exist no other delay and that all network elements are FIFO. Then the flow state $z^{\mathcal{H}_{\text{TAI}}}_{f,\text{S4}^*_{\text{East}}}$ of f at the output

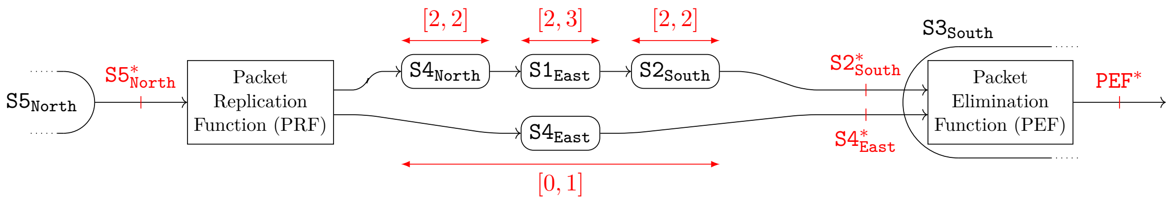


Figure 6.2: Flow graph $\mathcal{G}(f)$ (also identical to $\mathcal{G}(g)$) of the toy example from Chapter 4, last row of Table 4.2. The delay bounds in the vertices upstream of S3$_{\text{South}}$ are assumed to have been computed and their values are shown in red intervals showing the minimum and maximum delay bounds through a vertex, respectively.

of S4$_{\text{East}}$, observed with $\mathcal{H}_{\text{TAI}}$ is

$$z^{\mathcal{H}_{\text{TAI}}}_{f,\text{S4}^*_{\text{East}}} = \left(\gamma_{r_0,2b_0}, \left\{\begin{array}{r}\phi\\ \text{E5}_{\text{East}}\\ \text{S5}_{\text{North}}\\ \text{S4}_{\text{East}}\end{array}\right\}, \left\{\begin{array}{r}D[\phi]=1\\ D[\text{E5}_{\text{East}}]=1\\ D[\text{S5}_{\text{North}}]=1\\ D[\text{S4}_{\text{East}}]=0\end{array}\right\}, \left\{\begin{array}{r}d[\phi]=0\\ d[\text{E5}_{\text{East}}]=0\\ d[\text{S5}_{\text{North}}]=0\\ d[\text{S4}_{\text{East}}]=0\end{array}\right\}, \left\{\begin{array}{r}\lambda[\phi]=0\\ \lambda[\text{E5}_{\text{East}}]=0\\ \lambda[\text{S5}_{\text{North}}]=0\\ \lambda[\text{S4}_{\text{East}}]=0\end{array}\right\}\right) \tag{6.2}$$

In the above flow state, $D[\phi]$ upper-bounds the delay (measured with $\mathcal{H}_{\text{TAI}}$) of f between the source application (on top of the network stack) and v, whereas $D[\text{E5}_{\text{East}}]$ upper-bounds the delay between the output of the vertex E5$_{\text{East}}$ and v. In this example, we have assumed that there is no delay in the output port E5$_{\text{East}}$ or in its remote input port, so $D[\phi] = D[\text{E5}_{\text{East}}]$. Additionally, we note that $D[\text{S4}_{\text{East}}]$ equals 0 because it upper-bounds the delay between $u^* = \text{S4}^*_{\text{East}}$ and $v = \text{S4}^*_{\text{East}}$. Last, for any tag u, the system between u^* and v is FIFO thus $\lambda[u]$ equals 0.

The flow state $z^{\mathcal{H}_{\text{TAI}}}_{f,\text{S2}^*_{\text{South}}}$ of f at the output of vertex S2$_{\text{South}}$, observed with $\mathcal{H}_{\text{TAI}}$ is

$$z^{\mathcal{H}_{\text{TAI}}}_{f,\text{S2}^*_{\text{South}}} = \left(\gamma_{r_0,2b_0}, \left\{\begin{array}{r}\phi\\ \text{E5}_{\text{East}}\\ \text{S5}_{\text{North}}\\ \text{S4}_{\text{North}}\\ \text{S1}_{\text{East}}\\ \text{S2}_{\text{South}}\end{array}\right\}, \left\{\begin{array}{r}D[\phi]=7\\ D[\text{E5}_{\text{East}}]=7\\ D[\text{S5}_{\text{North}}]=7\\ D[\text{S4}_{\text{North}}]=5\\ D[\text{S1}_{\text{East}}]=2\\ D[\text{S2}_{\text{South}}]=0\end{array}\right\}, \left\{\begin{array}{r}d[\phi]=6\\ d[\text{E5}_{\text{East}}]=6\\ d[\text{S5}_{\text{North}}]=6\\ d[\text{S4}_{\text{North}}]=4\\ d[\text{S1}_{\text{East}}]=2\\ d[\text{S2}_{\text{South}}]=0\end{array}\right\}, \left\{\begin{array}{r}\lambda[\phi]=0\\ \lambda[\text{E5}_{\text{East}}]=0\\ \lambda[\text{S5}_{\text{North}}]=0\\ \lambda[\text{S4}_{\text{North}}]=0\\ \lambda[\text{S1}_{\text{East}}]=0\\ \lambda[\text{S2}_{\text{South}}]=0\end{array}\right\}\right) \tag{6.3}$$

We note that both $z^{\mathcal{H}_{\text{TAI}}}_{f,\text{S4}^*_{\text{East}}}$ and $z^{\mathcal{H}_{\text{TAI}}}_{f,\text{S2}^*_{\text{South}}}$ contain the same arrival curve: $\gamma_{r_0,2b_0}$. Indeed, the packets have suffered the same jitter (1 time unit) since their common source E5$_{\text{East}}$.

The concept of flow state replaces the concept of individual arrival curve in the TFA approach. In particular, if a vertex p *has been computed*, then the individual flow state $z_{f,p'}$ and z_{f,p^*} for each flow $f \ni p$ at p' and p^* has been obtained.

Identifying the Diamond Ancestors with the Tag Sets

The purpose of the tag set of a flow state is to ease the identification of diamond ancestors.

Assume for example that n can be computed and consider a flow $f \ni n$. Then the vertices that belong to the intersection of the tag sets of the flow states $\{z_{f,p^*}\}_{\forall \text{ parent } p}$ are diamond ancestors of n in $\mathcal{G}(f)$. If n contains a packet-elimination function (PEF) for f, then each identified diamond ancestor can be used to apply Item 2/ of Theorem 4.1.

Example: Consider the vertex $\texttt{S3}_{\texttt{South}}$. Its parents in $\mathcal{G}$ are $\texttt{S4}_{\texttt{East}}$ and $\texttt{S2}_{\texttt{South}}$. If we look at the tag sets in the flow states for f at the output of each parent ($z^{\mathcal{H}_{\text{TAI}}}_{f,\texttt{S4}^*_{\texttt{East}}}$ and $z^{\mathcal{H}_{\text{TAI}}}_{f,\texttt{S2}^*_{\texttt{South}}}$), we note that $\texttt{E5}_{\texttt{East}}$, $\texttt{S5}_{\texttt{North}}$ and the special tag ϕ are common to both sets. They are diamond ancestors of the vertex $\texttt{S3}_{\texttt{South}}$ in $\mathcal{G}(f)$. Each of them can be used for Item 2/ of Theorem 4.1 when computing the effect of $\texttt{PEF}_{\texttt{S3}_{\texttt{South}}}(f)$.

In the above Equations (6.2) and (6.3), the tag sets contain the list of *all* the upstream vertices. This is however not mandatory: a tag set can contain only a subset of them.

Example: Vertices $\texttt{S4}_{\texttt{North}}$, $\texttt{S1}_{\texttt{East}}$, $\texttt{S2}_{\texttt{South}}$ and $\texttt{S4}_{\texttt{East}}$ are “useless” tags in $z^{\mathcal{H}_{\text{TAI}}}_{f,\texttt{S4}^*_{\texttt{East}}}$ and $z^{\mathcal{H}_{\text{TAI}}}_{f,\texttt{S2}^*_{\texttt{South}}}$ because none of them is a diamond ancestor of $\texttt{S3}_{\texttt{South}}$ in $\mathcal{G}(f)$, thus removing them does not have any effect the application of Theorem 4.1.

Besides the special tag ϕ that remains mandatory in every tag set, the user can cherry-pick the vertices that are added to the tag set of a flow state. This can be done in order to keep track of only a few interesting ancestors. In its default setting for feed-forward networks, xTFA only adds a vertex a to the tag set if the vertex has several children in $\mathcal{G}(f)$.

Example: With the default setting of xTFA, both tag sets of $z^{\mathcal{H}_{\text{TAI}}}_{f,\texttt{S2}^*_{\texttt{South}}}$ and $z^{\mathcal{H}_{\text{TAI}}}_{f,\texttt{S2}^*_{\texttt{South}}}$ would only contain ϕ and $\texttt{S5}_{\texttt{North}}$. Indeed, $\texttt{S5}_{\texttt{North}}$ is the only vertex of $\mathcal{G}(f)$ that has two children in $\mathcal{G}(f)$. ϕ and $\texttt{S5}_{\texttt{North}}$ are diamond ancestors of $\texttt{S3}_{\texttt{South}}$. However, $\texttt{E5}_{\texttt{East}}$ is not in any of the two tag sets, despite the fact that it is a diamond ancestor of $\texttt{S3}_{\texttt{South}}$.

The default setting of xTFA is susceptible to allow only for the identification of a subset of diamond ancestors but it drastically reduces the memory size of the flow states in practice.

Changing the Observing Clock for a Flow State

Like delays, arrival curves or service surves, the flow state depends on the clock used to observe the flow, which we specify again by putting the clock in supper-script.

Changing the observing clock for a flow state is performed by Algorithm 3. It takes the flow state as observed with a clock $\mathcal{H}_i$ and the new clock $\mathcal{H}_g$ and returns a flow state for the same flow at the same observation point, as observed by the new clock $\mathcal{H}_g$.

In the algorithm, GetParamsTimeModel returns the parameters of the time model of Chapter 5: the clock-stability bound ρ, the time-jitter bound η and the synchronization precision Δ, with the convention that $\Delta = +\infty$ if the network is not synchronized.

ChangeClockArrivalCurve$(\alpha_1, \dots)$ applies Proposition 5.3 for changing the clock for an arrival curve with the respective time model. Formally, it returns the curve $\alpha_2 \in \mathfrak{F}_0$ with $\forall t > 0, \alpha_2(t) = \alpha_1(\min(\rho t+\eta, t+2\Delta))$. Its specific implementation depends on the underlying computer representation of arrival and service curves: it is provided by the xTFA min-plus back-end and consists in applying the sub-cases of Table 5.1.

Algorithm 3 Algorithm for changing the observing clock of a flow state

```
Require: $z_{f,v}^{\mathcal{H}_i}$: flow state of $f$ at $v$ as observed by $\mathcal{H}_i$. $\mathcal{H}_g$ new observing clock.
1:  procedure ChangeClockFlowState($z_{f,v}^{\mathcal{H}_i}, \mathcal{H}_g$)
2:      $(\alpha_1, \mathcal{U}, d_1, D_1, \lambda_1) \leftarrow z_{f,v}^{\mathcal{H}_i}$
3:      $(\rho, \eta, \Delta) \leftarrow$ getParamsTimeModel($\mathcal{H}_i, \mathcal{H}_g$)
4:      $\alpha_2 \leftarrow$ ChangeClockArrivalCurve($\alpha_1, \rho, \eta, \Delta$)        ▷ Proposition 5.3
5:      for $u \in \mathcal{U}$ do
6:          $d_2[u] \leftarrow \max(|d_1[u] - \eta|^+/\rho, d_1[u] - 2\Delta)$        ▷ Proposition 5.1
7:          $D_2[u] \leftarrow \min(\rho D_1[u] + \eta, D_1[u] + 2\Delta)$        ▷ Proposition 5.1
8:          $\lambda_2[u] \leftarrow \min(\lambda D_1[u] + \eta, \lambda_1[u] + 2\Delta)$        ▷ Proposition 5.1
9:      end for
10:     $z_{f,v}^{\mathcal{H}_g} \leftarrow (\alpha_2, \mathcal{U}, d_2, D_2, \lambda_2)$
11:     return $z_{f,v}^{\mathcal{H}_g}$
12: end procedure
```

6.2.3 The Computation Pipelines

The second major idea in xTFA is to provides more flexibility to the TFA model of each vertex, through the concept of *computation pipelines*.

Presentation of the Concept

Assume that a vertex n can be computed. Step 1 computes an arrival curve $\alpha_{n^\dagger}$ for the aggregate at the input of the CBQS using the individual flow states $\{z_{f,p'}, z_{f,p^*}\}_{p \text{ parent}, f \ni (p,n)}$.

In previous tools such as TFA++ [Mifdaoui, Leydier 2017] and FP-TFA (Chapter 3), this is performed by applying closed-form expressions that are adapted depending on the content of the vertex. For example, FP-TFA provides two closed-form expressions of the aggregate arrival curve: (3.5) when the vertex does not contain any regulator and (3.9) when it does.

With the addition of packet replication, elimination and ordering functions (PREOFs), obtaining a closed-form expression for the aggregate arrival curve by taking into account any possible configuration of the three optional functions (REGs, PEFs, POFs) is illusory. To remedy this issue, xTFA introduces the concept of *computation pipelines*.

Definition 6.2 (xTFA computation pipeline) *In xTFA, a computation pipeline is a chain of computational blocks, each of which takes as input a* pipeline data structure *and returns a modified version of the same data structure.*
Each vertex n contains three pipelines, corresponding to the three-step process*:*

- *an aggregate computation pipeline for computing the aggregate arrival curve $\alpha_{n^\dagger}$,*
- *a delay-bound computation pipeline for computing lower and upper bounds on the delay of the aggregate through the CBQS, and*
- *a flow-state computation pipeline for computing the individual output flow states $z_{f,n'}$ and z_{f,n^*} for each individual flow f crossing n.*

The organization of the pipelines in a vertex n is presented in Figure 6.3. For each pipeline

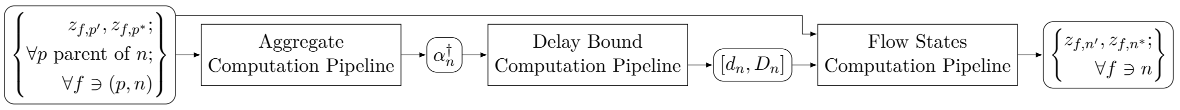

$z_{f,p'}, z_{f,p^*}$; $\forall p$ parent of n; $\forall f \ni (p,n)$
Aggregate Computation Pipeline
$\alpha_n^\dagger$
Delay Bound Computation Pipeline
$[d_n, D_n]$
Flow States Computation Pipeline
$z_{f,n'}, z_{f,n^*}$; $\forall f \ni n$

Figure 6.3: Organization of the three computation pipelines of a vertex n.

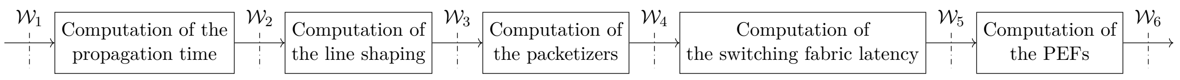

$\mathcal{W}_1$
Computation of the propagation time
$\mathcal{W}_2$
Computation of the line shaping
$\mathcal{W}_3$
Computation of the packetizers
$\mathcal{W}_4$
Computation of the switching fabric latency
$\mathcal{W}_5$
Computation of the PEFs
$\mathcal{W}_6$

Figure 6.4: Computational blocks in the *aggregate computation pipeline (ACP)* of vertex $\texttt{S3}_{\texttt{South}}$ in the toy example.

there exist a set of available computational blocks, each modeling a different mechanism, that follow a unified interface. Thus modeling a vertex n does not consist anymore in obtaining a closed-form expression for each step of the *three-step process*. It rather consists in selecting, for each pipeline, the right computational blocks, their parameters and their order, depending on the content of the vertex n and on the theoretical results that we want to use.

In the following, we focus on the aggregate computation pipeline (ACP) and detail its benefits using the toy example. The two remaining pipelines (computation of delay bounds and computation of individual flow states) are not presented in the manuscript because they are conceptually much simpler.

The Aggregate Computation Pipeline (ACP)

In an aggregate computation pipeline (ACP), each computational block receives an ACP data structure $\mathcal{W}$ and outputs a modified version of it, $\mathcal{W}'$. The content of an ACP data structure $\mathcal{W}$ is detailed later in the manuscript. For the moment we note that due to the unified interface, the blocks can be removed and/or moved to follow the model of a specific vertex.

Example: Figure 6.4 presents the content of the ACP of vertex $\texttt{S3}_{\texttt{South}}$ in the toy example. The first block, “Computation of the transmission links”, receives an ACP data structure $\mathcal{W}^1$ and outputs $\mathcal{W}^2$, a modified version of it.

If $\texttt{S3}_{\texttt{South}}$ does not contain any PEF, then the block *Computation of the PEFs* can simply be removed. If $\texttt{S3}_{\texttt{South}}$ contains regulators either before or after the PEFs, then a corresponding block *Computation of the REGs* can be added to its ACP at the correct location.

To ease the notation, we now define the Observation Points points 1,3,4,5,6 on the upper-part of Figure 6.1. The “Observation Point 1” is placed on top of $\texttt{S2}'_{\texttt{South}}$ and $\texttt{S4}'_{\texttt{South}}$. It represents the union of the two observation points $\texttt{S2}'_{\texttt{South}}$ and $\texttt{S4}'_{\texttt{South}}$, in the sense that an observer placed at Observation Point 1 counts both the bits crossing both $\texttt{S2}'_{\texttt{South}}$ and $\texttt{S4}'_{\texttt{East}}$.

Therefore, the cumulative function $R_{f,1}$ of f at Observation Point 1 is $R_{f,1} = R_{f,\texttt{S2}'_{\texttt{South}}} + R_{f,\texttt{S4}'_{\texttt{East}}}$. From Definition 2.5, $R_{f,1}$ is constrained by $\alpha_{f,\texttt{S2}'_{\texttt{South}}} + \alpha_{f,\texttt{S4}'_{\texttt{East}}}$. The same apply for $R_{g,1}$ and for R_1, the cumulative arrival of the aggregate.

As we will later detail, there exists a relationship between the data structure W_j in Figure 6.4 (corresponding to the output of the j-th computational block) and the arrival curve for the aggregate at Observation Point j.

The role of the ACP is hence to model the effect of each mechanism/phenomenon on the aggregate arrival curve, step by step.

From the Origins of Line Shaping to the Design of the ACP Data Structure $\mathcal{W}$

In our general network-calculus model for TSN networks (Chapter 2), we model each transmission link as an element with a fixed propagation delay, followed by a greedy shaper (Section 2.6.5). This greedy shaper formalizes the line-shaping effect effect that has been introduced in [Grieu 2004].

In his seminal work, Grieu states that, for a vertex n, if we focus on the stream of packets $\{f|f \ni (p,n)\}$ that come from the same parent p, then this aggregate cannot cross the upstream physical link within p faster than the capacity of the physical link, c_p. In [Grieu 2004], the aggregate $\{f|f \ni (p,n)\}$ of the flows coming from the same parent p is called a *group* and the shaping curve γ_{c_p} of the physical link within p is called its *group envelope*.

Taking into account the line shaping significantly reduces the delay bounds. Many TFA-based tools include the line-shaping effect in their closed-form expressions (Chapter 3, [Bouillard 2022, Algorithm 2], [Mifdaoui, Leydier 2017, Eq. 3]).

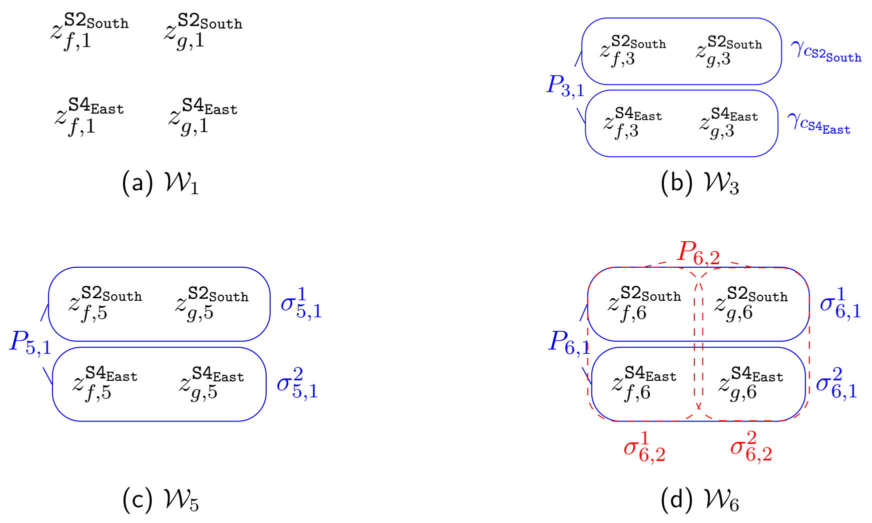
In xTFA, we do not want to use closed-form expressions. Therefore, to ensure the flexibility of the ACP, its data structure $\mathcal{W}$ shall be able to capture the effects of the different mechanisms, including the line shaping. Thus the design of the ACP data structure $\mathcal{W}$ is greatly inspired by the seminal ideas of Grieu with *groups* and *group envelopes*.

Definition 6.3 (ACP data structure) *An **ACP data structure** $\mathcal{W}$ is a pair $\mathcal{W} \triangleq (\mathcal{S},\mathcal{P})$ that contains:*

- *$\mathcal{S}$, a set of* flow states *(defined in Section 6.1)*
- *$\mathcal{P}$, a set of partitions of $\mathcal{S}$ such that, for each partition $P \in \mathcal{P}$, each partition element e of P is noted $e = (Z,\sigma)$ and represents a* group *of flow states Z together with a piecewise-linear concave arrival curve σ, called the* group envelope.

Our definition contains three major differences with respect to [Grieu 2004, §3.2]. First, it is more flexible because it allows for multiple partitions. Then, it continues to provide the raw list of flow states (hence of arrival curves) for each individual flow. This property removes the issue mentioned by Grieu in [Grieu 2004, §3.2.3.2]. Last, our ACP data structure is only local to a node and is destroyed (saving memory space) as soon as the aggregate arrival curve $\alpha_{n^\dagger}$ has been obtained, unlike in [Grieu 2004, §3.2.3.1].

A computation block of an aggregate computation pipeline (ACP) is therefore an object that receives an ACP data structure $\mathcal{W}$ and returns a modified version of it, $\mathcal{W}'$. Any operation on $\mathcal{W}$ is authorized: adding or removing flows states from $\mathcal{S}$, modifying any field of any flow state of $\mathcal{S}$, adding or removing partitions from $\mathcal{P}$, and modifying any partition of $\mathcal{P}$ by removing, adding or modifying any of its groups. Overall, the final xTFA tool provides a dozen computational blocks for modeling the line shaping, the technological latencies, the regulators, the internal sources, the PREOFs functions, etc.



$z^{\mathtt{S2_{South}}}_{f,1}$ $z^{\mathtt{S2_{South}}}_{g,1}$
$z^{\mathtt{S4_{East}}}_{f,1}$ $z^{\mathtt{S4_{East}}}_{g,1}$
(a) $\mathcal{W}_1$
$P_{3,1}$
$z^{\mathtt{S2_{South}}}_{f,3}$ $z^{\mathtt{S2_{South}}}_{g,3}$
$\gamma_{c_{\mathtt{S2_{South}}}}$
$z^{\mathtt{S4_{East}}}_{f,3}$ $z^{\mathtt{S4_{East}}}_{g,3}$
$\gamma_{c_{\mathtt{S4_{East}}}}$
(b) $\mathcal{W}_3$
$P_{5,1}$
$z^{\mathtt{S2_{South}}}_{f,5}$ $z^{\mathtt{S2_{South}}}_{g,5}$
$\sigma^1_{5,1}$
$z^{\mathtt{S4_{East}}}_{f,5}$ $z^{\mathtt{S4_{East}}}_{g,5}$
$\sigma^2_{5,1}$
(c) $\mathcal{W}_5$
$P_{6,2}$
$P_{6,1}$
$z^{\mathtt{S2_{South}}}_{f,6}$ $z^{\mathtt{S2_{South}}}_{g,6}$
$\sigma^1_{6,1}$
$z^{\mathtt{S4_{East}}}_{f,6}$ $z^{\mathtt{S4_{East}}}_{g,6}$
$\sigma^2_{6,1}$
$\sigma^1_{6,2}$ $\sigma^2_{6,2}$
(d) $\mathcal{W}_6$

Figure 6.5: Content of the ACP data structure at several observations points of Figure 6.1.

6.2.4 Modeling the Line Shaping and the Packet Elimination Function Using ACP Computation Blocks

In this subsection, we highlight the flexibility offered by the ACP and the ACP data structure by detailing the computational blocks that model the propagation time, the line shaping and the packet-elimination function (PEF).

To ease with the explanation, we assume that all clocks are ideal, equal to $\mathcal{H}_{\mathrm{TAI}}$, and we consequently omit the clock super-script for the different notions. We use the super-script to distinguish the provenance of a flow state: z^p_f denotes the flow state for the stream of packets that belong to f and enter n from p. We use the lower-script to distinguish the observation point (as in the rest of the thesis): $z^p_{f,v}$ denotes the flow state for the stream of packets of f coming from p, observed at v. When v is a number j (as in $z^p_{f,1}$, with $j = 1$), it denotes the value of the flow state as written in $\mathcal{W}_i$, *i.e.*, after the i-th computational step in the pipeline.

Initialization of the ACP data structure $\mathcal{W}$:

The structure $\mathcal{W}$ is initialized with the flow states at p', the output of the previous CBQS for any parent p. Formally, the data structure is initialized with $\mathcal{W}_1 = (\mathcal{S}_1, \mathcal{P}_1)$ with $\mathcal{S}_1 = \{z^p_{h,p'}; \forall p \text{ parent of } n; \forall h \ni (p,n)\}$ and $\mathcal{P}_1 = \emptyset$ is empty as no groups have been formed yet.

Example: $\mathcal{S}_1 = \{z^{\mathtt{S2_{South}}}_{f,1}, z^{\mathtt{S2_{South}}}_{g,1}, z^{\mathtt{S4_{East}}}_{f,1}, z^{\mathtt{S4_{East}}}_{g,1}\} \triangleq \{z^{\mathtt{S2_{South}}}_{f,\mathtt{S2}'_{\mathtt{South}}}, z^{\mathtt{S2_{South}}}_{g,\mathtt{S2}'_{\mathtt{South}}}, z^{\mathtt{S4_{East}}}_{f,\mathtt{S4}'_{\mathtt{East}}}, z^{\mathtt{S4_{East}}}_{g,\mathtt{S4}'_{\mathtt{East}}}\}$ is the set of the flow states for f and g at the output of the CBQSs of each of the two parents $\mathtt{S2_{South}}$ and $\mathtt{S4_{East}}$. Figure 6.5a shows a graphical representation of the data-structure $\mathcal{W}_1$ with $z^p_{h,1} \triangleq z^p_{h,p'}$ for $h \in \{f,g\}$ and $p \in \{\mathtt{S2_{South}}, \mathtt{S4_{East}}\}$. Flow states are distributed in the plane and no association of them into groups is present so far ($\mathcal{P}_1 = \emptyset$).

Computational Block for the Propagation Time of the Transmission Links

The first computation block in the pipeline (Figure 6.4) is responsible for computing the effect of the propagation time in the transmission links. This is performed by Algorithm 4.

Algorithm 4 Algorithm updating the data-structure $\mathcal{W}$ to model the propagation time of the transmission links.

```
Require: $\forall p$, $T_p^{\text{prop}}$ is the propagation delay (assumed constant and expressed in $\mathcal{H}_{\text{TAI}}$) of
    the transmission link within vertex $p$. $\mathcal{W}_1 = (\mathcal{S}_1, \mathcal{P}_1)$ is the input ACP data structure.
 1: procedure ComputeEffectOfPropagationTime($\mathcal{W}_1$)
 2:     $(\mathcal{S}_1, \mathcal{P}_1) \leftarrow \mathcal{W}^1$
 3:     $\mathcal{S}_2 \leftarrow \emptyset$
 4:     for $z_{h,1}^p \in \mathcal{S}_1$ do
 5:         $(\alpha_1, \mathcal{U}_1, D_1, d_1, \lambda_1) \leftarrow z_{h,1}^p$
 6:         for $u \in \mathcal{U}_1$ do
 7:             $D_2[u] \leftarrow D_1[u] + T_p^{\text{prop}}$
 8:             $d_2[u] \leftarrow d_1[u] + T_p^{\text{prop}}$
 9:         end for
10:         $z_{h,2}^p \leftarrow (\alpha_1, \mathcal{U}_1, D_2, d_2, \lambda_1)$
11:         add $z_{h,2}^p$ to $\mathcal{S}_2$
12:     end for
13:     $\mathcal{W}_2 \leftarrow (\mathcal{S}_2, \mathcal{P}_1)$
14:     return $\mathcal{W}_2$
15: end procedure
```

As the propagation delay is assumed constant, it does not affect the arrival curves (Lemma B.3) or worsen the RTO [Mohammadpour, Le Boudec 2021, Thm.7]. Therefore, for each flow state $z_{h,1}^p$ in the input data structure $\mathcal{W}_1$, Algorithm 4 creates a flow state $z_{h,2}^p$ for $\mathcal{W}_2$ with the same arrival curve, the same tag set and the same RTO map (Line 10). The minimum and maximum delay-bound maps for the new flow state are simply increased by the propagation delay T_p^{prop} of the respective parent p (Lines 8 and 7).

Example: The result $\mathcal{W}_2$ of Algorithm 4 on the toy example is almost identical to $\mathcal{W}_1$, but with different flow states $z_{h,2}^p$ for $h \in \{f, g\}, p \in \{\texttt{S2}_{\texttt{South}}, \texttt{S4}_{\texttt{East}}\}$.

Computational Block for the Line Shaping

The line shaping is computed by using Algorithm 5 that follows the idea in [Grieu 2004]. It creates a new partition, $P_{3,1}$ on Line 3. For each parent p of n in $\mathcal{G}$, it then creates a partition element that contains all flow states coming from the parent p (Line 5) and this *group* is associated with the *group envelope*, or shaping curve γ_{c_p} (Line 6). The group and the group envelope are then added to the partition $P_{3,1}$ (Line 7).

We note that the set of flow states $\mathcal{S}_2$ has not been modified and is directly transmitted to $\mathcal{W}_3$. Indeed, σ_{c_p} is sub-additive, hence the greedy shaper keeps the arrival-curve constraints and does not increase the end-to-end latency bounds (Theorem 2.4).

Algorithm 5 Algorithm updating the data-structure $\mathcal{W}$ to take into account the line shaping (greedy shaper of the transmission links).

```
Require: $n$ the node to compute, $\mathcal{G}$ the network graph.
Require: $\forall q$ vertex of $\mathcal{G}$, $c_q$ is the capacity (assumed constant and expressed in $\mathcal{H}_{\text{TAI}}$) of the
    transmission link within $p$.
 1: procedure ComputeEffectOfLineShaping($\mathcal{W}_2$)
 2:     $(\mathcal{S}_2, \mathcal{P}_2) \leftarrow \mathcal{W}_2$
 3:     $P_{3,1} \leftarrow$ empty partition
 4:     for $p$ a parent of $n$ in $\mathcal{G}$ do
 5:         $Z \leftarrow \{z^q_{h,2} \in \mathcal{S}_2 | q = p\}$
 6:         $\sigma \leftarrow \gamma_{c_p}$
 7:         add $(Z, \sigma)$ to $P_{3,1}$
 8:     end for
 9:     $\mathcal{P}_3 \leftarrow \mathcal{P}_2 \cup \{P_{3,1}\}$
10:     $\mathcal{W}_3 \leftarrow (\mathcal{S}_2, \mathcal{P}_3)$
11:     return $\mathcal{W}_3$
12: end procedure
```

Example: The resulting data structure $\mathcal{W}_3$ is shown in Figure 6.5b. It contains the flow states that are the same as in $\mathcal{W}_2$: $\forall h \in \{f, g\}, \forall p \in \{\texttt{S2}_{\texttt{South}}, \texttt{S4}_{\texttt{East}}\}, z^p_{h,3} = z^p_{h,2}$. But $\mathcal{W}_3$ additionally has a partition of the flow states ($P_{3,1}$, in solid blue lines) with two groups, one per input port, each having its group envelope.

Note that this graphical representation depicts an aggregate arrival curve

$$\alpha_3 = \left(\alpha^{\texttt{S2}_{\texttt{South}}}_{f,3} + \alpha^{\texttt{S2}_{\texttt{South}}}_{g,3}\right) \otimes \gamma_{c_{\texttt{S2}_{\texttt{South}}}} + \left(\alpha^{\texttt{S4}_{\texttt{East}}}_{f,3} + \alpha^{\texttt{S4}_{\texttt{East}}}_{g,3}\right) \otimes \gamma_{c_{\texttt{S4}_{\texttt{East}}}} \tag{6.4}$$

obtained by summing the individual arrival curves among a group $(\alpha^{\texttt{S2}_{\texttt{South}}}_{f,3} + \alpha^{\texttt{S2}_{\texttt{South}}}_{g,3})$, performing the min-plus convolution with the group envelope, $\otimes\gamma_{c_{\texttt{S2}_{\texttt{South}}}}$ and summing this result with the results from the other groups in the partition. α_3 is an arrival curve for the aggregate at Observation Point 3 in Figure 6.1.

Computational Blocks for the Packetization and The Switch Fabric Latency

Let us now jump directly to $\mathcal{W}_5$ by providing only a few comments on the next two computational blocks in Figure 6.4. The computational block for the packetizer applies Theorems 3.1 and 2.2 to each individual flow, as well as to each aggregate defined by a group in a partition. Hence, it modifies individual arrival curves, as well as group envelopes, but it does not modify the content of the groups within the partitions. Additionally, it does not increase the latency bounds in the flow states, because the packetizer does not increase the per-packet delay bounds (Theorem 2.5).

The computational block that models the technological latencies in the input port and the switch fabric increases the lower [resp., the upper] delay bounds in all flow states by the minimum [resp., maximum] traversal time of the switching fabric. All arrival curves (both individual ones and group envelopes) are worsened by applying Theorem 2.2 because any input/output pair of the switching fabric is assumed causal, FIFO and lossless (Section 2.6.3).

Example: The output of the two steps, $\mathcal{W}_5$, is presented in Figure 6.5c. The flow states $z^p_{h,5}$ differ from $z^p_{h,3}$ because of the packetizer and of the latency in the switching fabric. $P_{5,1}$ contains the same groups as in $P_{3,1}$, but the group envelopes $\sigma^1_{5,1}$ and $\sigma^2_{5,1}$ differ from the original group envelopes $\gamma_{\texttt{cs2}_\texttt{South}}$ and $\gamma_{\texttt{cs4}_\texttt{East}}$ in $\mathcal{W}_3$ due to the packetization and to the switch fabric latency.

Computational Block for the Packet Elimination Function

The effect of the packet-elimination functions (PEFs), computed with the tight model of Thoerem 4.1 is last obtained using Algorithm 6. This algorithm creates a new partition, $P_{6,2}$ that contains a group for each eliminated flow.

Algorithm 6 Algorithm updating the data-structure $\mathcal{W}$ to take into account the packet-elimination function (Theorem 4.1)

```
Require: $n$ the node to compute, $\mathcal{G}$ the network graph.
 1: procedure ComputeEffectOfPEFs($\mathcal{W}_5$)
 2:     $(\mathcal{S}_5, \mathcal{P}_5) \leftarrow \mathcal{W}_5$
 3:     $P_{6,2} \leftarrow$ empty partition
 4:     for $h$ crossing $n$ such that $n$ contains a PEF for $h$ do
 5:         $\mathcal{Z} \leftarrow \{z^p_{5,h} \in \mathcal{S}_5; \forall p$ parent of $n$ in $\mathcal{G}\}$
 6:         $\mathcal{A} \leftarrow \bigcap\{\mathcal{U}; \forall(\alpha, \mathcal{U}, D, d, \lambda) \in \mathcal{Z}\}$
                              ▷ Common tags are diamond ancestors of $n$ in $\mathcal{G}(h)$
 7:         $\sigma \leftarrow \delta_0$
 8:         for $a \in \mathcal{A}$ do
 9:             $d_h^{a\to n} \leftarrow \min\{d[a]; \forall(\alpha, \mathcal{U}, D, d, \lambda) \in \mathcal{Z}\}$
10:             $D_h^{a\to n} \leftarrow \max\{D[a]; \forall(\alpha, \mathcal{U}, D, d, \lambda) \in \mathcal{Z}\}$
11:             $\alpha_{h,a^*} \leftarrow$ GetArrivalCurve($\mathcal{G}$,$a^*$,$h$)
12:             $\alpha_f^{a\to n} \leftarrow \alpha_{h,a^*} \oslash \delta_{D_h^{a\to n} - d_h^{a\to n}}$        ▷ Item 2/ of Theorem 4.1
13:             $\sigma \leftarrow \sigma \otimes \alpha_f^{a\to n}$                                   ▷ Theorem 4.1
14:         end for
15:         add group $(\mathcal{Z}, \sigma)$ to $P_{6,2}$
16:     end for
17:     $\mathcal{O} \leftarrow$ all remaining flow states
18:     add group $\left(\mathcal{O}, \sum_{(\alpha,\mathcal{U},d,D)\in\mathcal{O}} \alpha\right)$ to $P_{6,2}$
19:     $\mathcal{P}_6 \leftarrow \mathcal{P}_5 \cup \{P_{6,2}\}$
20:     $\mathcal{W}_6 \leftarrow (\mathcal{S}_5, \mathcal{P}_6)$
21:     return $\mathcal{W}_6$
22: end procedure
```

For each eliminated flow h, Line 5 first selects $\mathcal{Z}$, the subset of $\mathcal{S}_4$ that contains all the flow states $z^p_{h,5}$ of flow h, for any parent p. Line 6 then intersects all the tag sets contained in the flow states of $\mathcal{Z}$. This intersection $\mathcal{A}$ is a subset of the diamond ancestors of n in $\mathcal{G}(h)$.

Then, the group envelope is initialized with the neutral element for the min-plus convolution at Line 7. For each identified diamond ancestor $a \in \mathcal{A}$, Line 9 computes the minimum of the values $d[a]$ for all the maps d in all the flow states in $\mathcal{Z}$. Note that for any map $d[]$ in a flow state of $\mathcal{Z}$, $d[a]$ exists because $a \in \mathcal{A} \subset \mathcal{U}$ for any tag set $\mathcal{U}$ of a flow state of $\mathcal{Z}$. By definition of the maps $d[]$, and because $\mathcal{Z}$ contains all the flow states of h, the minimum of the $d[a]$ is a lower-bound on the minimum delay for h between the output of a and the input of the PEF, along any possible path $a \to n$, *i.e.*, it is the value $d_f^{a\to n}$ defined in Theorem 4.1. Similarly, Line 10 obtains $D_f^{a\to n}$.

Line 11 then queries from the network graph $\mathcal{G}$ the arrival curve that the flow h had at the output of a, α_{h,a^*}. In a feed-forward network, this query is always valid because a is a ancestor of n thus the *three-step* process has necessarely been peformed for a before we reach this line of the algorithm for n. Line 12 of Algorithm 6 applies Item 2/ of Theorem 4.1 with ancestor a to obtain $\alpha_f^{a\to n}$, an arrival curve for the flow h (*i.e.*, for the aggregate made of all its flow states) at the output of the PEF. The arrival curve is finally combined with the one from other ancestors (Line 13) by applying Theorem 4.1.

Before we add $P_{6,2}$ to the list of partitions in the new data-strcuture $\mathcal{W}_6$, we need to make sure that it is a partition across all the flow states that belong to $\mathcal{S}_6 = \mathcal{S}_5$. Therefore, the subset $\mathcal{O}$ of $\mathcal{S}_5$ that contains all the flow states $z_{h,5}^p$ for the flows h for which the current vertex n does not have a PEF are grouped into a single last group, whose group envelope in simply the sum of their individual arrival curves (Line 18).

Finally, Algorithm 6 can add the new partition $P_{6,2}$ without modifying neither the flow states ($\mathcal{S}_6 = \mathcal{S}_5$) nor the previous partitions because any set of parallel PEFs is FIFO and without any delay thus it keeps the delay bound and the arrival curves, both the individual arrival curves and those of the aggregates (Lemma B.3).

Example: The final result $\mathcal{W}_6$ of the ACP for the toy example is given in Figure 6.5d. The four flow states have not changed compared to $\mathcal{W}_5$ (for each h, p, $z_{h,6}^p = z_{h,5}^p$), it is a consequence of Item 1/ of Theorem 4.1. Similarly, the previously-existing partition $P_{5,1}$ (continuous blue lines) is kept ($P_{6,1} = P_{5,1}$) because aggregate arrival curves are kept by a set of parallel PEFs, as the latter have no delay (Lemma B.3).

A new partition $P_{6,2}$ (dashed red lines) has been added by Algorithm 6. Here again, its graphical representation depicts an aggregate arrival curve

$$\alpha_{6,2} = \left(\alpha_{f,6}^{\mathtt{S2}_{\mathtt{South}}} + \alpha_{f,6}^{\mathtt{S4}_{\mathtt{East}}}\right) \otimes \sigma_{6,2}^1 + \left(\alpha_{g,6}^{\mathtt{S2}_{\mathtt{South}}} + \alpha_{g,6}^{\mathtt{S4}_{\mathtt{East}}}\right) \otimes \sigma_{6,2}^2 \tag{6.5}$$

which is equivalent to the application of Theorem 4.1 for both f and g. $\sigma_{6,2}^1$ is a convolution of the arrival curves $\alpha_f^{a_1\to n} \otimes \alpha_f^{a_2\to n} \otimes \dots$ for the subset of diamond ancestors $a_1, a_2, \dots$ that are identified trough the intersection of the tag sets in the flow states of f (Line 6 of Algorithm 6). When the flow states for f are as in Equations (6.2) and (6.3), then the intersection of the tag sets reveals that ϕ, $\mathtt{E5}_{\mathtt{East}}$ and $\mathtt{S5}_{\mathtt{North}}$ are diamond ancestors of n in $\mathcal{G}(f)$, thus $\sigma_{6,2}^1 = \alpha_f^{\phi\to n} \otimes \alpha_f^{\mathtt{E5}_{\mathtt{East}}\to n} \otimes \alpha_f^{\mathtt{S5}_{\mathtt{North}}\to n}$. With the default configuration of xTFA for feed-forward networks however, $\mathtt{E5}_{\mathtt{East}}$ would not have been identified in the intersection, hence $\sigma_{6,2}^1$ would have been only equal to $\alpha_f^{\phi\to n} \otimes \alpha_f^{\mathtt{S5}_{\mathtt{North}}\to n}$ and the result is less tight but remains valid.

The above example highlights that the ACP data structure $\mathcal{W}$ is able to model both effects (line shaping and packet elimination). It thus ensures the flexibility of the aggregate computation pipeline and allows for interchangeable and movable computational blocks.

To obtain the final aggregate arrival curve $\alpha_{n^\dagger}$ at the input of the CBQS, we apply Algorithm 7 on the final data-structure $\mathcal{W}_6$. As intuited in the toy example, for each partition

P of the data-structure, Algorithm 7 computes an aggregate arrival curve following the groups of the partition: On Line 7, the arrival curve $\alpha_{(Z,\sigma)}$ for a group (Z,σ) of the partition P is obtained by the convolution of the group envelope σ with the sum of the individual arrival curves for the members of the group Z. On Line 8, the arrival curve for all groups in the partition P are summed to finally obtain α_P, an arrival curve for the aggregate, computed as per partition P.

Because each partition P gives an aggregate arrival curve α_P, they can be combined together and their convolution (Line 10) gives an arrival curve for the aggregate. This convolution $\alpha_{\text{aggregate}}$ is initialized on Line 3 with the sum of all individual arrival curves, which correspond to the trivial partition in which each flow state is alone in its own group.

Algorithm 7 Algorithm for obtaining an arrival curve for the aggregate by using the data structure $\mathcal{W}$

```
1: procedure DataStructureToAggregateArrivalCurve($\mathcal{W}$)
2:     $(\mathcal{S},\mathcal{P}) \leftarrow \mathcal{W}$
3:     $\alpha_{\text{aggregate}} \leftarrow \sum_{(\alpha,\mathcal{U},D,d)\in\mathcal{S}} \alpha$
4:     for $P \in \mathcal{P}$ do
5:         $\alpha_P \leftarrow 0$
6:         for $(Z,\sigma) \in P$ do
7:             $\alpha_{(Z,\sigma)} \leftarrow \left(\sum_{(\alpha,\mathcal{U},D,d)\in Z} \alpha\right) \otimes \sigma$
8:             $\alpha_P \leftarrow \alpha_P + \alpha_{(Z,\sigma)}$
9:         end for
10:        $\alpha_{\text{aggregate}} \leftarrow \alpha_{\text{aggregate}} \otimes \alpha_P$
11:    end for
12:    return $\alpha_{\text{aggregate}}$
13: end procedure
```

6.3 Adaptations of xTFA for Cyclic Dependencies

In feed-forward networks, xTFA computes end-to-end latency bounds through the iterative application of the three pipelines for each vertex n, following a topological sort of the GIF.

In networks with cyclic dependencies, we face the same issue as in FP-TFA: The vertices cannot be ordered through a topological sort. To compute latency bounds in such networks we transform the real network into a virtual feed-forward network using cuts. We then rely on the extended fixed-point theorem of Chapter 4 (Theorem 4.3). The theorem states that if $\mathcal{FF} : (\boldsymbol{b},\boldsymbol{d}) \mapsto (\boldsymbol{b}',\boldsymbol{d}')$ is an application that maps the bursts and upper delay bounds at the input of the virtual feed-forward network into the burst and delay bounds at the output of the virtual feed-forward network; if the real network is empty at $t = 0$; and if $(\overline{\boldsymbol{b}},\overline{\boldsymbol{d}})$ is a finite non-negative fixed-point of $\mathcal{FF}$, then the real network is stable and $(\overline{\boldsymbol{b}},\overline{\boldsymbol{d}})$ represent valid burst and delay bounds at the cut locations in the real network.

With xTFA, the data structure that bounds the properties of a flow f at an observation point v is the flow state $z_{f,v}$. Hence, the iterative computation of the nodes in the virtual feed-forward network provides an application $\mathcal{XX} : \boldsymbol{z} \mapsto \boldsymbol{z'}$ that maps the flow states after the cuts to the flow states before the cuts. To use Theorem 4.3, we must prove the equivalence of $\mathcal{XX}$ and $\mathcal{FF}$. It requires a set of restrictions on xTFA that we list hereafter.

6.3.1 Restrictions on xTFA for Networks with Cyclic Dependencies

When xTFA is used on networks with cyclic dependencies, the following restrictions are applied:
– First, as with FP-TFA, only leaky-bucket individual arrival curves are supported, for each flow f and each observation point v, $\alpha_{f,v} = \gamma_{r_f, b_{f,v}}$ with r_f the rate of flow f and $b_{f,v}$ the burst of the flow at v.
– Second, for every flow state $z_{f,v}$, the only tag authorized in the tag set is ϕ, the source application of f. With this restriction, the second item of Theorem 4.1 (PEF output arrival curve) can only be applied with the diamond ancestor $a = \phi$.
– Third, the RTO bound $\lambda[\phi]$ is removed from any flow states. As a consequence, this restricted version of xTFA cannot perform any computation that would require the knowledge of this RTO bound. For example, this restricted version of xTFA does not support packet-ordering function (POF) when placed without a following regulator, because computing the effect of such POF on the arrival curve requires the RTO [Mohammadpour, Le Boudec 2021, Cor. 1]. The restricted version can still compute delay bounds when POFs are immediately followed by REGs, because the arrival curve at the output of the POF does not need to be known for computing the contention in the CBQS placed after the REGs.

As a consequence of the above limitations, each flow state $z_{f,v}$ can be written $z_{f,v} = (\gamma_{r_f, b_{f,v}}, \phi, d[\phi], D[\phi])$ with $b_{f,v}$ the burst of f at v, $d[\phi]$ [resp., $D[\phi]$] a lower-bound on the minimum delay [resp., an upper-bound on the maximum delay] for f between ϕ and v.

6.3.2 Latency Bounds in Networks With Cyclic Dependencies

We note $\boldsymbol{z} = \{z_i\}_{i\in[\![1,p]\!]}$ the vector of flow states after the cuts (*i.e.*, at the input of the feed-forward network) and $\boldsymbol{z}' = \{z'_i\}_{i\in[\![1,p]\!]}$ the vector of flow states after the cuts (*i.e.*, at the output of the feed-forward network), with p the size of the two vectors.

Then the application $\mathcal{XX}$ obtained by the computation of the virtual feed-forward network with the restricted version of xTFA can be written

$$\mathcal{XX} : \{z_i = (\gamma_{r_i,b_i}, \phi, D_i[\phi], d_i[\phi])\}_{i\in[\![1,p]\!]} \mapsto \left\{z'_i = (\gamma'_{r_i}, b'_i, \phi, D'_i[\phi], d'_i[\phi])\right\}_{i\in[\![1,p]\!]} \tag{6.6}$$

In this formulation, for any $i \in [\![1,p]\!]$, r'_i and r_i are both equal to the rate $r_{f,\phi}$ of the flow f at its source ϕ, where f is the flow corresponding to the flow state z_i. Similarly, $d_i[\phi]$ and $d'_i[\phi]$ are equal because the lower-bound on the delay can be computed offline and does not depend on the contention within the nodes. Therefore, removing all constants, $\mathcal{XX}$ can be written

$$\mathcal{XX} : (b_i, D_i[\phi])_{i\in[\![1,p]\!]} \mapsto (b'_i, D'_i)_{i\in[\![1,p]\!]} \tag{6.7}$$

Hence, $\mathcal{XX}$ obtained with the restricted xTFA on the virtual feed-forward network is equivalent to the application $\mathcal{FF}$ of Theorem 4.3. If the restricted xTFA finds a finite non-negative fixed-point[3] $\mathcal{X}\mathcal{X}(\overline{z}) = \overline{z}$, then the real network with the cyclic dependencies is stable and the constituents of $\overline{z}$ represent valid upper-bounds for the bursts and delays at the cuts.

Conclusion

In this chapter we have described experimental modular TFA (xTFA). xTFA describes a set of original data structures and algorithms for computing end-to-end latency bounds in networks with cyclic dependencies, redundancy mechanisms and non-ideal clocks (synchronized or not). It also describes an experimental implementation of these algorithms based on Python [20].

In Section 6.1 we provided an overview of the tool with its requirements and main features. Then in Section 6.2 we have discussed the main ideas: the *flow states*, the *computation pipelines* and the *flow-state partitions* within the data-structure of the aggregate computation pipeline (ACP). We have provided the algorithms for the computational blocks that model the line-shaping effect and the redundancy mechanisms. We have also highlighted the concepts by using a toy example from the previous chapters.

Finally in Section 6.3, we have discussed how the extended fixed-point result of Chapter 4 (Theorem 4.3) is applied in the context of xTFA. In particular we have seen that it leads to several restrictions on the capacities of the tool, with respect to the computations of feed-forward networks.

xTFA implements the majority of our theoretical contributions for computing performance upper-bounds. However, we also provide in this thesis several results that prove the inexistence of such performance upper-bounds in some situations: Theorem 4.5 when an IR is placed after a PEF, Proposition 5.5 when regulators are non-adapted in a non-synchronized network, Proposition 5.10 when an IR is non-adapted in a synchronized network. These results can also be confirmed through the use of simulations, because simulations lower-bound the true worst-case. In the following chapter, we discuss how the ns-3 network simulator has been modified to validate that the IR can yield unbounded latencies when placed non-adapted in a network with non-ideal but synchronized clocks.

[3] In the tool, we add a quantification step in the third pipeline of every node so that the delay bounds and the burst bounds are written as multiple of a fixed resolution. Hence a strict equality of the upper-bounds is possible.

Chapter 7

Implementation of Local Clocks in ns-3

"We have so long had only ourselves to fight that we are used to such internecine quarrels. An invader that finds us divided against ourselves will dominate us all, or destroy us all. The only true defense is to produce Galaxia, which cannot be turned against itself [. . .]."

Golan Trevize

Isaac Asimov, *Foundation and Earth.*

Contents

7.1 ns-3, a Network Simulator 138
7.1.1 A Discrete Event Simulator 139
7.1.2 Nodes, NetDevices and Channels 139
7.1.3 The ns-3 Aggregation System 139
7.1.4 The Context and the Simulator Global Variable 141
7.2 Previous Work, Limitations and New Requirements 142
7.2.1 Managing Local Clocks in Discrete Event Simulators (DESs) 142
7.2.2 Previous Attempts for Local Clocks in ns-3 144
7.2.3 Specifications for a New Attempt at Implementing ns-3 Local Clocks . . . 144
7.3 Proposed Design for the Implementation of Clocks in ns-3 145
7.3.1 Overview of the Solution 146
7.3.2 Scheduling an Event 146
7.3.3 Updating a Clock 148
7.3.4 Validation of the Design Requirements 148
7.4 Application: Simulating the IR Instability with Non-Ideal Clocks . . 150

In Chapter 5, Proposition 5.10 states that a non-adapted IR placed after a FIFO system can yield unbounded latencies if three sources with different clocks are placed before the FIFO system. Unlike positive results, this negative result is not proved using the concepts of network calculus: it exhibits a trajectory with adversarial sources and adversarial clocks that yields unbounded latency. The proof in Appendix B.3.10 is long because we need to explicit all the consequences of the choices of the adversaries. Yet, this length can hinder the diffusion of our results and makes the proof hard to peer-review. Therefore, to add even more credit to the validity of Proposition 5.10, we decided to simulate the adversarial behavior and validate that it leads indeed to unbounded latencies.

From February to June 2020, Guillermo Aguirre Rodrigo performed a master project at EPFL with the goal of simulating the proposed trajectory and checking that the delays are indeed unbounded. The project was carried out under the supervision of Professor Le Boudec and with advice from this thesis' author.

Guillermo quickly identified two options for the simulator choice: OMNET++ [17] with its add-on NeSTINg [Falk, *et al.* 2019] and ns-3 [16]. At that time, none of them had the support for simulating clock non-idealities[1] or interleaved regulators[2].

The simulator ns-3 was selected because (a) the NeSTINg project did not appear to be well maintained [11], (b) the ns-3 documentation is much more consistent than the documentation of NeSTINg or INET, and (c) this thesis' author had prior experience with ns-3. Therefore, Guillermo's project had three objectives: (1) allow for the support of local clocks in ns-3; (2) add the model of TSN ATS (the implementation of IR within TSN) to ns-3; and (3) configure the clock models, the source behavior and the ATS parameters as in the proof of Proposition 5.10 and analyze the delay bounds. The author of this thesis co-designed with Guillermo the solution for the first objective. This solution is presented in this chapter as we believe that it is of interest beyond Guillermo's master project. A merge request based on this work has been opened on the ns-3 mainline code and is under discussion with the ns-3 maintainers for integration [1].

We first present the key concepts of ns-3 in Section 7.1. Then the previous work for simulating local clocks in discrete-event simulators (DESs) is discussed in Section 7.2. From their limitations we derive a set of requirements for the solution. The proposed solution is then presented in Section 7.3 Finally, its application for the adversarial case described in the proof of Proposition 5.10 is discussed.

This manuscript does not discuss the other two objectives in which Guillermo led alone the conception of the solutions. For them, we refer to his report [Aguirre Rodrigo 2020].

7.1 ns-3, a Network Simulator

ns-3 [16] is an open-source object-oriented discrete-event simulator (DES) written in C++ for modeling and simulating networks and Internet systems. Its has been widely used for research and educational use and has received the 2020 SIGCOMM Networking Systems Award that recognizes "the development of a networking system that has had significant impact on the world of computer networking" [19].

As opposed to many other network simulators (OMNET++ [17], Optnet [18], ...), ns-3 does not come with a default graphical user interface (GUI) and the simulation scenarios are written in either C++ or Python scripts. However, the ns-3 web-page offers a thorough tutorial and a complete documentation. The simulator offers a very high level of flexibility, allowing for an easy implementation of new models. In this section, we present some of the

[1]After the end of the project, a support for simulating clocks has been added into INET [4], a framework on which NeSTINg relies. This recent implementation of clocks in INET is discussed in Section 7.2.

[2]The NeSTINg respository [11] has an open merge request for ATS, the TSN implementation of IRs, but as of June 2022, the merge request has not been either updated or merged since September 2019.

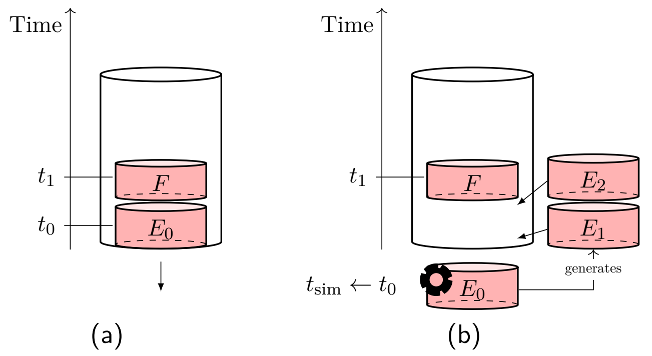


Figure 7.1: Principle of the event scheduler in a discrete-event simulator (DES). (a) The events are sorted as per their execution time. (b) The simulator picks the event with the smallest execution instant and 'jumps' to this time instant by updating its simulation time t_{sim}. The execution of the event can generate new events. (c) New events are inserted in the event scheduler as per their execution time instants.

key concepts of ns-3 on which our implementation of local clocks relies.

7.1.1 A Discrete Event Simulator

A discrete-event simulator (DES) maintains a list of to-be-executed events, ordered as per their execution time instants in the simulation time. In ns-3, this list is called the event scheduler. At each step, the DES processes the event with the smallest time instant by "jumping" directly to this time instant, *i.e.*, updating the simulation time to the execution time of the event. It then executes the actions associated with this event. The execution of an event can result in new events being inserted at various time instants in the scheduler.

Example: Assume that event F is scheduled at time t_1, after event E_0, as in Figure 7.1a. When E_0 is executed, it generates two new events E_1 and E_2 as in Figure 7.1b. If their execution time instants are before the execution time of F as in Figure 7.1c, they are inserted before F.

7.1.2 Nodes, NetDevices and Channels

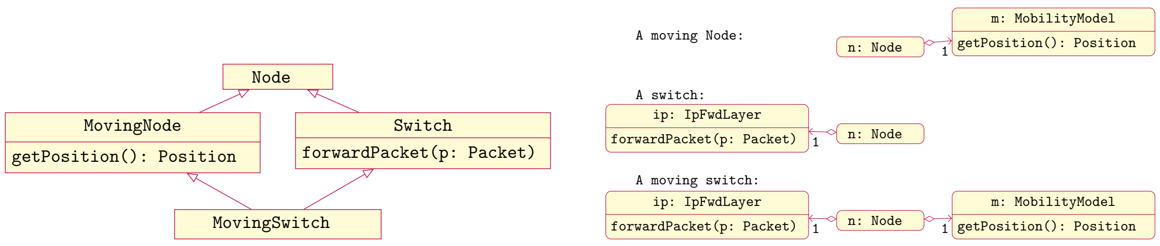
The base element in an ns-3 simulation is the `Node`. A `Node` can have several networking devices, or `NetDevices`, that are connected to other `NetDevices` in other `Nodes` through `Channels`. One of the simplest types of `Channels` in ns-3 is the `PointToPointChannel` that is connected to exactly two `NetDevices` and offers a full duplex, collision-free link between them, which corresponds to two of our (one-way) "transmission links" used throughout the thesis. A comparison of the terms used in ns-3 with those used throughout the thesis is shown in Table 7.1. Each `Node` in ns-3 has a unique identifier, its `node-id`.

7.1.3 The ns-3 Aggregation System

The ns-3 aggregation system is one on the most important concepts in ns-3 and our implementation of local clocks heavily relies on it. At its origin, the aggregation system is an elegant solution to the following problem:

Table 7.1: Comparison of the terms used in the thesis with the ns-3 terminology.

Term used in ns-3	Term used in the thesis
A `Node`	A “Device”
A `PointToPointChannel`	Two “transmission links” (one per direction)
A `NetDevice`	An “output port” + An “input port”



(a) In ns-2: a `Node` is specialized by sub-classing the base class `Node`.

(b) In ns-3: a `Node` is specialized by aggregating objects to it.

Figure 7.2: Illustration of the solutions embraced by ns-2 and ns-3 to specialize the `Node`s and distinguish between switches, moving nodes, etc. (a) UML class diagram for ns-2: ns-2 uses the traditional sub-classing paradigm to specialize the `Node`s. (b) UML object diagram for ns-3: ns-3 uses an aggregation system in which the model of a `Node` is modified by aggregating objects to it, for example a moving node is an instance of `Node` that has a `MobilityModel`.

Since all elements in the simulation are ns-3 `Node`s, how do we distinguish between `Node`s that are end systems and those that are bridges, or switches ? How do we distinguish between the `Node`s that have an IP forwarding layer, those that move, or not ?

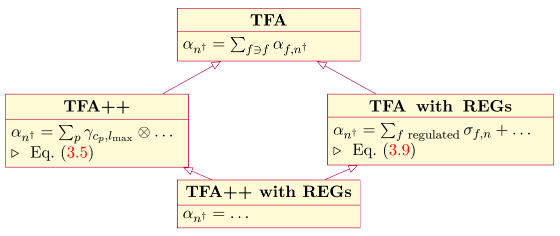
In object-oriented computer programming, the classic solution for adding or overriding a behavior of a base class (the `Node`) is to create a subclass of it: For example, we could create a subclass `MovingNode` for a moving `Node`, a subclass `Switch` for an IP switch, etc. This solution, used by the ns-2 network simulator, is represented in Figure 7.2a. As discussed in the ns-3 documentation [16, Manual, Section “Aggregation”], it raises several issues.

For example, if we need to model a `Node` that is both a `Switch` and a `MovingNode`, then we would create a class that inherits from both `MovingNode` and `Switch` (Figure 7.2a). This is not possible in all object-oriented programming languages and for those that allow it (e.g., C++), it raises an ambiguity known as the “diamond problem” [15].

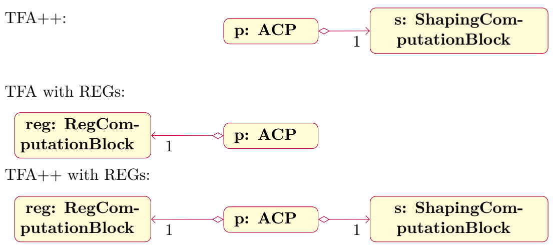
In ns-3, there exist no sub-classes of `Node`. Instead, the behavior of a `Node` instance is modified by aggregating different objects to it. Figure 7.2b illustrate some examples[3]: A moving node is an instance of `Node`, to which we aggregate a `MobilityModel` object. A switch is a `Node` to which we aggregate an `IpFwdLayer`. And a moving switch is a `Node` to which we aggregate both a `MobilityModel` and an `IpFwdLayer`.

The aggregation system of ns-3 is such that any ns-3 object (e.g., an instance of `Mobility-Model`) can be aggregated to any other ns-3 object of different type (e.g., an instance of `Node`), by using a simple call `n->AggregateObject(o)`. The only limitation is that only one object

[3]The UML diagrams in Figure 7.2 are only for illustration purposes, they do not represent the exact conception in ns-2 or ns-3. Note that the rounded boxes in Figure 7.2b represent instances and not classes. Indeed, the aggregations in Figure 7.2b are not coded in the class but created during run-time.


TFA
$\alpha_{n^\dagger} = \sum_{f \ni f} \alpha_{f,n^\dagger}$
TFA++
$\alpha_{n^\dagger} = \sum_p \gamma_{c_p,l_{\max}} \otimes \dots$
▷ Eq. (3.5)
TFA with REGs
$\alpha_{n^\dagger} = \sum_{f \text{ regulated}} \sigma_{f,n} + \dots$
▷ Eq. (3.9)
TFA++ with REGs
$\alpha_{n^\dagger} = \dots$

(a) Illustrative class diagram with FP-TFA.



(b) Illustrative object diagram with xTFA.

Figure 7.3: Illustration of the solutions embraced by FP-TFA and xTFA for distinguishing between vertex models with or without line shaping, with or without regulators, etc. (a) FP-TFA uses the traditional sub-classing paradigm to overwrite the closed-form expression of the aggregate arrival curve. (b) xTFA use the concept of aggregate computation pipelines (ACPs) where each pipeline p is specified by aggregating computation blocks to it.

per type can be aggregated: for example, only one `MobilityModel` can be aggregated to a given `Node`.

If we have a pointer `n` to an aggregate and if we know the type of the aggregated object that we are interested in, then `n->GetObject<Type>()` returns a pointer to the (unique) aggregated object of actual type `Type`. For example, if `n` is a pointer to a moving `Node`, then `n->GetObject<MobilityModel>()` returns a pointer to its (unique) `MobilityModel` and `n->GetObject<MobilityModel>()->GetPosition()` returns the node's position.

Remark: The ns-3 aggregation system is very powerful because the aggregation is performed during run-time. The system has greatly inspired the conception of xTFA, presented in details in Chapter 6. Indeed, in a TFA approach, the model for each vertex n has a part that computes the aggregate arrival curve $\alpha_{n^\dagger}$ at the observation point $n^\dagger$. Thus xTFA faces the following challenge: How do we distinguish between the vertices on which we take into account the line shaping effect and those on which we do not ? Those that have packet-elimination functions, or regulators and those that have no optional function ? Figure 7.3 highlights two solutions to this problem: In FP-TFA (Figure 7.3a), the computation of the aggregate arrival curve is overwritten depending on the vertex model. In xTFA, the computation of the arrival curve is performed by the aggregate computation pipeline (ACP), to which we aggregate the required pieces of model.

7.1.4 The Context and the Simulator Global Variable

In ns-3, each event executed by the DES has a `context`, whose value can be obtained from anywhere in the ns-3 code. This integer value is equal to the `node-id` of the `Node` on which the event takes place[4]. With this `node-id` we can obtain a pointer to the `Node` using `NodeList::GetNode(node-id)`, where `NodeList` is a global variable.

The `Simulator` global variable (accessible from anywhere in the source code) is the entry point for scheduling events. It provides the main following function:

– `Simulator::Schedule(D, action, object, arguments)` takes four parameters:

[4]Except in special cases, e.g., during the setup of the simulation.

- `D` is the delay between the current simulation time and the time of the scheduled event.
- `action` is the action to be executed (a pointer to a function).
- `object` is the pointer to the C++ object that should execute the above function.
- `arguments` is a list of arguments to provide to the above function.

`Simulator::Schedule(D, action, object, arguments)` schedules a new event in the event scheduler, that should be executed at `t+D`, where `t` is the current simulation time. When the new event is to be executed, the scheduler calls `object->action(arguments)`. The object behind the `Simulator` must inherit from the `SimulatorImpl` interface.

ns-3 provides a default implementation for the interface: `DefaultSimulatorImpl`. With this implementation, there exists only one time reference frame: the simulation time, which corresponds to the international atomic time (*temps atomique international*) (TAI) in our thesis, noted $\mathcal{H}_{\mathrm{TAI}}$. The delay `D` that is provided to the `Schedule` function is understood as a delay observed with $\mathcal{H}_{\mathrm{TAI}}$. The simulator uses a unique event scheduler, with the events being ordered as per their execution time in $\mathcal{H}_{\mathrm{TAI}}$ and the events are executed one after the other (`DefaultSimulatorImpl` does not support mutli-threading).

During the setup of the simulation, the user can change the default simulator for a different implementation, as long as this implementation inherits from the `SimulatorImpl` interface.

7.2 Previous Work, Limitations and New Requirements

The idea of supporting different local clocks in discrete-event simulators (DESs) is not new. However, it realization requires to address two important challenges:
– First, the architecture of the DES must be adapted. Instead of scheduling all the events in the same scheduler and with the same clock, the DES shall manage different clocks, identify the clock that is used to schedule an event, synchronize the events across different clocks, etc.
– Second, a model for the local clocks shall be designed and implemented in the simulator. As discussed in Chapter 4, the stochastic properties of the clocks have been widely studied in the literature. However, creating a computer-representation of such clock models that is suited for use in DESs is challenging [Renczes, Kovácsházy 2020; Mahmood, *et al.* 2017]. A method for implementing a realistic clock model is presented[5] in [Gaderer, *et al.* 2011].

In the thesis, we focus on the first challenge: extending the current capabilities of ns-3 to allow for discrete-event simulation with local clocks, where each local clock has an unspecified model that is hidden behind an application programming interface (API).

7.2.1 Managing Local Clocks in Discrete Event Simulators (DESs)

In [Zhu, Ma, Ryu 2013], a "System and method for clock modeling in discrete-event simulation" is patented. In the invention, when an event E is scheduled at a time instant $t^{\mathcal{H}}$ observed

[5]For the interested reader, we should note that, contrary to what the authors indicate in [Gaderer, *et al.* 2011, §2] , the ns-3 simulator does not include any linear model for local clocks. The `WallClockSynchronizer` class cited by the authors is in reality an utility class (not supposed to be used by the user) that is used by the `RealtimeSimulatorImpl`. The latter allows for the user to run the simulation "in real time", *i.e.*, synchronized with the computer's clock (the "wall clock"). Neither `WallClockSynchronizer` nor `RealtimeSimulatorImpl` are intended to simulate local clocks.

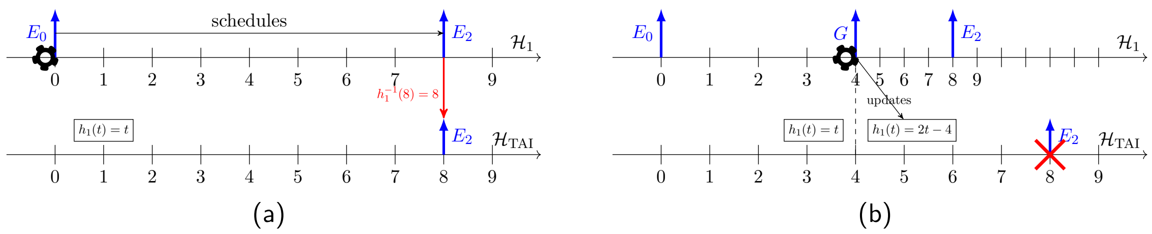


Figure 7.4: Principles and issues of [Zhu, Ma, Ryu 2013]. (a) Principle of the patent: when an event E_2 is scheduled with a local clock at $t_{E_2}^{\mathcal{H}_1}$, its execution time is converted into the global time using $h_1^{-1}(t_{E_2}^{\mathcal{H}_1})$, and inserted into the unique event scheduler. (b) An issue arises if the clock model (the function h_1) is updated by another event G before E_2 could have been executed. Because of this update, the previous execution time of E_2 in the global event scheduler is not valid anymore.

with a local clock $\mathcal{H}$, the simulator first translates the time instant into $t^{\mathcal{H}_{\text{TAI}}}$ observed with the true time $\mathcal{H}_{\text{TAI}}$ using the API of the model for clock $\mathcal{H}$. Then the event E is inserted in the event scheduler at time $t^{\mathcal{H}_{\text{TAI}}}$, with the rest of the events.

Example: In Figure 7.4a, the execution of Event E_0 schedules the new event E_2 at $t_{E_2}^{\mathcal{H}_1} = t + T = 8$ time units. At the execution of event E_0, the mapping between $\mathcal{H}_{\text{TAI}}$ and $\mathcal{H}_1$ is $h_1(t) = t$: The local clock is perfectly synchronized with the true time.

With the technique of [Zhu, Ma, Ryu 2013], the execution time of E_2 is translated into $\mathcal{H}_{\text{TAI}}$ using the clock model API: $t_{E_2}^{\mathcal{H}_{\text{TAI}}} = h_1^{-1}(t_{E_2}^{\mathcal{H}_1}) = 8$ time units. And event E_2 is inserted into the unique event scheduler at execution time $t_{E_2}^{\mathcal{H}_{\text{TAI}}} = 8$ time units.

With the solution in [Zhu, Ma, Ryu 2013], the simulator only needs to maintain a unique event scheduler and a unique timeline: the events are sorted in the event scheduler as per their execution time instants observed in $\mathcal{H}_{\text{TAI}}$. The local clocks $\{\mathcal{H}_i\}$ are only used for creating the events and for converting time instants/durations from/to the local/global time. Our conception of local clocks in ns-3 relies on these basic principles[6].

A major issue with the technique of [Zhu, Ma, Ryu 2013] arises when the clock model is updated: the mapping between global time and local time changes.

Example: Assume that at $t^{\mathcal{H}_{\text{TAI}}} = 4$, event G is executed (Figure 7.4b). For example, G can correspond to the reception of a synchronization message that is processed by the synchronization client within Node 1. This event could not have been anticipated when E_0 was executed. As a consequence of the synchronization message, the synchronization client decides to steer the node's clock, for example by doubling its frequency.

Then event E_2 is still to be executed at $t_{E_2}^{\mathcal{H}_1} = 8$ time units, but since the mapping $h_1(t)$ has changed, $t_{E_2}^{\mathcal{H}_{\text{TAI}}}$ now equals 6 time units instead of 8 previously. The technique of [Zhu, Ma, Ryu 2013] only keeps track of a unique scheduler aligned with the $\mathcal{H}_{\text{TAI}}$ timeline (the bottom timeline in Figure 7.4). In this scheduler, event E_2 is not at the correct execution time, it must be rescheduled.

[6]At the beginning of the project, we were not aware of the existence of [Zhu, Ma, Ryu 2013]. We found it natural to convert all the events from their local clock to the true time and to manage all the events in a unique event scheduler whose clock is $\mathcal{H}_{\text{TAI}}$. We also found it natural that the clock models should convert time instants and durations from the local clock to the true time and reciprocally, which leads to the API described in [Zhu, Ma, Ryu 2013].

On the 31st of July, 2020, around a month after Guillermo opened a merge request for submitting our solution of local clocks to the ns-3 community, the community of the open-source INET framework [8] also published their own implementation of local clocks for the OMNET++ simulator [4]. Their implementation shows some similarities with the design proposed in this chapter.

For example, they also rely on the patent of [Zhu, Ma, Ryu 2013] and schedule all events in a unique event scheduler [9, Lines 47-70]. Like in our design, their implementation stores the events scheduled with a clock in a list [10, Lines 91, 98] and uses this list to reschedule the events if the clock model is updated [10, Lines 129-156]. However, as opposed to our design, their implementation is not backwards compatible: Only the models that have been implemented after their implementation of local clocks and that explicitly use the call `scheduleClockEventAt` of their implementation (instead of the default call `scheduleAt`) benefit from their implementation.

7.2.2 Previous Attempts for Local Clocks in ns-3

In the context of ns-3, several works have been previously conducted for implementing local clocks in ns-3. None of them has been integrated in the ns-3 mainline code.

Coudron and Secci developed a first approach for implementing local clocks in ns-3 [Coudron, Secci 2015]. In their design, the `Node` class is modified to implement the `Simulator` interface. Therefore each node i in the simulation has its own event scheduler in which time instants are observed with $\mathcal{H}_i$. The event schedulers are synchronized through a global scheduler (observed with $\mathcal{H}_{\text{TAI}}$) that keeps track of only the next-to-be-executed event for each node. In an email [14], Coudron argues that the solves avoids the problems raised by the update of a clock model. However, their proposal is not backwards compatible because the events need to be scheduled with `n->Schedule()` instead of `Simulator::Schedule()`, where `n` is a pointer to the node of the current context.

Another project based on ns-3 has been presented in [Maruyama, *et al.* 2015]. The authors evaluate the performance of the [IEEE 1588] synchronization protocol using ns-3 simulations. To this end, they develop a clock model and a subclass of the ns-3 `Application` class: `ApplicationOnClock`. Their design follows the overall design of the patent [Zhu, Ma, Ryu 2013]. The authors also address the issue raised when updating a clock model: when a clock model for $\mathcal{H}_i$ is updated, all the events that have been scheduled using $\mathcal{H}_i$ are removed from the event scheduler, their new time instants in $\mathcal{H}_{\text{TAI}}$ are computed and the events are inserted again in the scheduler, at the corrected time instants in $\mathcal{H}_{\text{TAI}}$.

The design of [Maruyama, *et al.* 2015] is not backwards compatible: Only the models that subclass `ApplicationOnClock` benefits from the clock model.

7.2.3 Specifications for a New Attempt at Implementing ns-3 Local Clocks

Guillermo and this thesis' author relied on both the ideas and the limitations of the projects presented above to determine a set of requirements that the implementation of local clocks in ns-3 should meet.

1. The implementation shall permit the definition of at least one local clock per ns-3 `Node`.

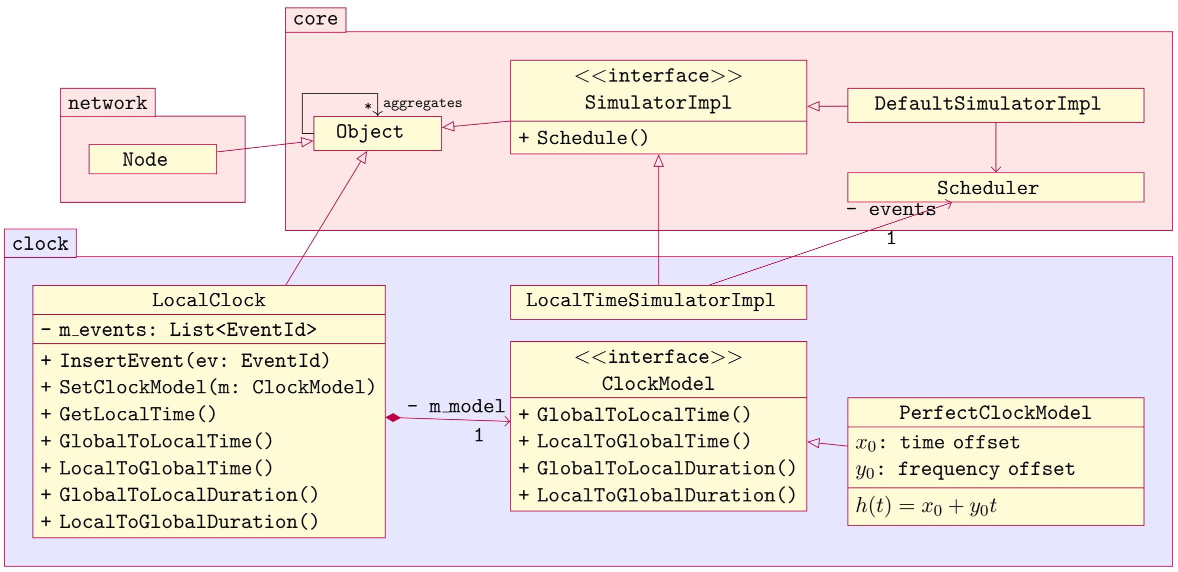


Figure 7.5: UML class diagram of the design of local clocks in ns-3
Unified Modeling Language (UML) class diagram of the design of local clocks in ns-3.

2. The implementation shall define an API that the future clock models must implement. This API shall be complete[7] and minimal[8]. For illustration purposes, the implementation will provide an easy clock model that implements said API. However, implementing realistic clock models is not the scope of the project.

3. The implementation shall provide an interface that allows for an external application to update the clock model during run-time (for example if an IEEE 1588 application receives a synchronization message, it may use this interface to steer the parameters of the clock model).

4. The implementation shall be backwards compatible (all previously-coded ns-3 models shall benefit from the local-clock implementation) and come as a totally independent module that does not require to change any line in the previously-existing models[9].

7.3 Proposed Design for the Implementation of Clocks in ns-3

In this section, the solution developed by Guillermo Aguirre and this thesis' author, and implemented by the former, is presented.

7.3.1 Overview of the Solution

The solution is presented in Figure 7.5, in which the new module `clock` is detailed, as well as its interactions with already-existing modules of ns-3 (`network` and `core`). The already-existing modules in red are not modified by the proposed approach. The solution is made of two classes: `LocalTimeSimulatorImpl` and `LocalClock` and one interface, `ClockModel`. An example of implementation of the interface `ClockModel` is also provided: `PerfectClockModel` represents a "perfect" clock with no noise and two parameters: the time offset x_0 and the frequency offset y_0: The local time $h(t)$ at true time t is $h(t) = x_0 + t \cdot y_0$.

The class `LocalTimeSimulatorImpl` implements the ns-3 interface `SimulatorImpl`. As such, `LocalTimeSimulatorImpl` can be instantiated and can take the role of `Simulator` everywhere in the simulation, in place of the `DefaultSimulatorImpl`. After the setup, all the calls to `Simulator::Schedule` are sent to `LocalTimeSimulatorImpl` instead of the default implementation. Like most of the related work, `LocalTimeSimulatorImpl` relies on an event scheduler in which the events are ordered as per their time instants observed with $\mathcal{H}_{\text{TAI}}$.

The interface `ClockModel` represents the minimal API that a clock shall provide in order to be integrated in the simulation. This API provides four services: converting a local time instant from the local clock to the global clock $\mathcal{H}_{\text{TAI}}$ and vice versa; and similarly for durations.

The class `LocalClock` represents a local clock. It relies on a clock model that shall inherit from `ClockModel`. The attribute `m_model` is private such that only the `LocalClock` has access to the model. The only way for other classes to change the clock model is through the `SetClockModel` method of `LocalClock`. The `LocalClock` class mirrors the four services of the model's API to the external classes. Note that `LocalClock` inherits from `Object` and as such it can be aggregated to any other `Object` of different type. In particular, a `LocalClock` can be aggregated to a `Node` to model the fact that the `Node` has a local clock. The `LocalClock` also keeps track in its private list (`m_events`) of all the events that have been scheduled with the current clock model and that have not been executed yet. This list is used when the clock model is updated through a call to `LocalClock::SetClockModel`.

In the following, we provide more insights to the proposed design by detailing two use-cases: when an event is scheduled and when the clock model is updated.

7.3.2 Scheduling an Event

The simplified sequence diagram for the first use-case is shown in Figure 7.6. Here we assume that the simulator executes an event (the `runningEvent` on the far-left). This running event schedules a new event to be executed in D seconds from now.

To do so, the model[10] calls `Simulator::Schedule(`D`,...)` and provides the delay D as well as the function and parameters to be executed by the scheduled event. This call is delivered to `sim`, the configured instance for `SimulatorImpl`.

[7]Everything that the simulator needs from the clock model in order to schedule and manage the events shall be written in the API.

[8]Nothing more than what is strictly necessary for the simulator to schedule and manage the events shall be requested from the clock model.

[9]This requirement permits a faster acceptation of the implementation in the community, as it guarantees that the behavior of the simulator is not changed when the local-clock module is not used.

[10]It can be any anything of the ns-3 source code: the model of an application, of the TCP layer, etc.

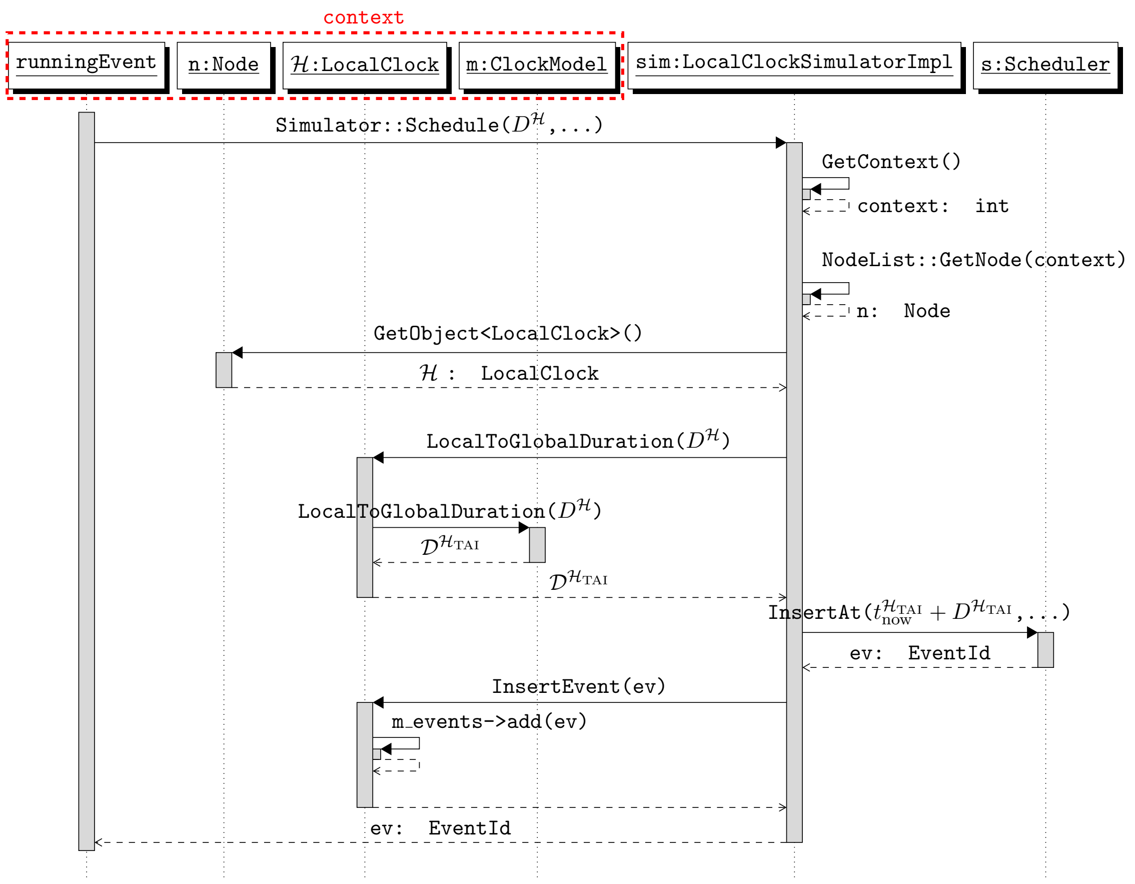

context
runningEvent
n:Node
$\mathcal{H}$:LocalClock
m:ClockModel
sim:LocalClockSimulatorImpl
s:Scheduler
Simulator::Schedule($D^{\mathcal{H}}$,...)
GetContext()
context: int
NodeList::GetNode(context)
n: Node
GetObject<LocalClock>()
$\mathcal{H}$: LocalClock
LocalToGlobalDuration($D^{\mathcal{H}}$)
LocalToGlobalDuration($D^{\mathcal{H}}$)
$D^{\mathcal{H}_{\text{TAI}}}$
$D^{\mathcal{H}_{\text{TAI}}}$
InsertAt($t_{\text{now}}^{\mathcal{H}_{\text{TAI}}} + D^{\mathcal{H}_{\text{TAI}}}$,...)
ev: EventId
InsertEvent(ev)
m_events->add(ev)
ev: EventId

Figure 7.6: Sequence diagram of the proposed design for the use-case "Scheduling a new Event". The `runningEvent` uses `Simulator::Schedule` to schedule a new event. This call is forwarded to `sim`, an instance of `LocalClockSimulatorImpl` that takes care of identifying the clock observing the delay $D^{\mathcal{H}}$ and for converting the delay to a delay observed with $\mathcal{H}_{\text{TAI}}$. Last, the event is inserted in the `Scheduler`.

If `sim` were an instance of the ns-3 default simulator, *i.e.*, `SimulatorDefaultImpl`, it would directly insert this event in the event scheduler at execution time instant $t_{\text{now}} + D$, where t_{now} is the current time. But since `sim` is an instance of `LocalSimulatorImpl`, it does not interpret D as being observed with the global time $\mathcal{H}_{\text{TAI}}$. Instead, it interprets the provided delay as being a delay $D^{\mathcal{H}}$ observed with a local clock $\mathcal{H}$.

The first step is to find the clock $\mathcal{H}$ that is used to observe the provided delay $D^{\mathcal{H}}$. The simulator queries the context, *i.e.*, `node-id`, of the `runningEvent` using the `GetContext` call. It then obtains a pointer `n` to the node by using `NodeList::GetNode(context)`. Last, `sim` uses the ns-3 aggregation system to obtain a pointer to the `LocalClock` aggregated to the `Node` by calling `GetObject<LocalClock>()` on `n`.

Now that `sim` has found a pointer to the `LocalClock` $\mathcal{H}$, it request $\mathcal{H}$ to convert the local delay $D^{\mathcal{H}}$ into a global delay $D^{\mathcal{H}_{\text{TAI}}}$ by calling `LocalToGlobalDuration`. $\mathcal{H}$ forwards this call to its private `ClockModel` and returns $D^{\mathcal{H}_{\text{TAI}}}$. The simulator `sim` now inserts the event in the scheduler, at the global time instant $t_{\text{now}}^{\mathcal{H}_{\text{TAI}}} + D^{\mathcal{H}_{\text{TAI}}}$, where $t_{\text{now}}^{\mathcal{H}_{\text{TAI}}}$ is the current global time.

Before returning the `EventId ev` to the calling `runningEvent`, `sim` notifies the `Local-Clock` $\mathcal{H}$ that the event has been created. $\mathcal{H}$ inserts this event in its personal list of events.

This list is useful for the next use-case. It does not need to be ordered.

7.3.3 Updating a Clock

We now consider the use-case in which an external model needs to update the clock model. In Figure 7.7, the external PTP client model receives a synchronization message from the ns-3 simulation. It decides to steer the clock parameters accordingly. To do so, it cannot directly set the parameters of the private attribute `model` of the local clock. Instead, it creates a new instance `new` of `ClockModel` (using the sub-class that fits its needs), configures it and uses `LocalClock::SetClockModel(new)` to set it as the new model of the local clock.

When `LocalClock::SetClockModel(new)` is called, the `LocalClock` sets its new model, but it also reschedules all the events of its personal event list that have not yet been executed.

Therefore, for each event `ev` in its personal list, the `LocalClock` requests from the `Simulator` the duration $D_{\text{old}}^{\mathcal{H}_{\text{TAI}}}$ between $t_{\text{now}}^{\mathcal{H}_{\text{TAI}}}$ and $t_{\text{old}}^{\mathcal{H}_{\text{TAI}}}$, observed with $\mathcal{H}_{\text{TAI}}$. This gives the remaining duration before the event, in $\mathcal{H}_{\text{TAI}}$, when the mapping between $\mathcal{H}$ and $\mathcal{H}_{\text{TAI}}$ is as per the `old` clock model. If $D_{\text{old}}^{\mathcal{H}_{\text{TAI}}}$ is positive, `LocalClock` requests the `old` clock model to translate this remaining delay in local time. This gives $D^{\mathcal{H}}$, the remaining delay before the event, observed with $\mathcal{H}$, and this delay does not depend on the model for $\mathcal{H}$.

Then, the `LocalClock` requests the `LocalTimeSimulatorImpl` to cancel (not execute) the event `ev`, because its execution time in the event scheduler is incorrect. Ideally, the `Scheduler` class would provide a method for removing an event from its queue. This is unfortunately not the case in the ns-3 code of the `Scheduler` as of today. A workaround as well as other related challenges are discussed on the merge request [1].

Last but not least, the `LocalClock` requests the `LocalTimeSimulatorImpl` to schedule a copy of the event in $D^{\mathcal{H}}$ seconds from now. To do so, it uses the `Simulator::Schedule` call whose implementation is discussed in the previous subsection. During the execution of `Simulator::Schedule`, the `ClockModel` to which the `LocalClock` forwards the `LocalTo-GlobalDuration` is the new clock model because `LocalClock` has previously updated its attribute `m_model`.

7.3.4 Validation of the Design Requirements

Through the two use cases, we observe that the proposed solution meets the requirements that are listed in Section 7.2.3. Indeed: (1) One `LocalClock` object can be aggregated per `Node`. (2) The design defines an API for the clock models, and provide a simple implementation, `PerfectClockModel`. (3) Through the call `LocalClock::SetClockModel`, an external model can steer the parameters of the clock model and all the events remained scheduled at the correct time instant. (4) All the already-existing the calls to `Simulator::Schedule` are automatically redirected to `LocalTimeSimulatorImpl` (if configured), without requiring to change any already-existing code.

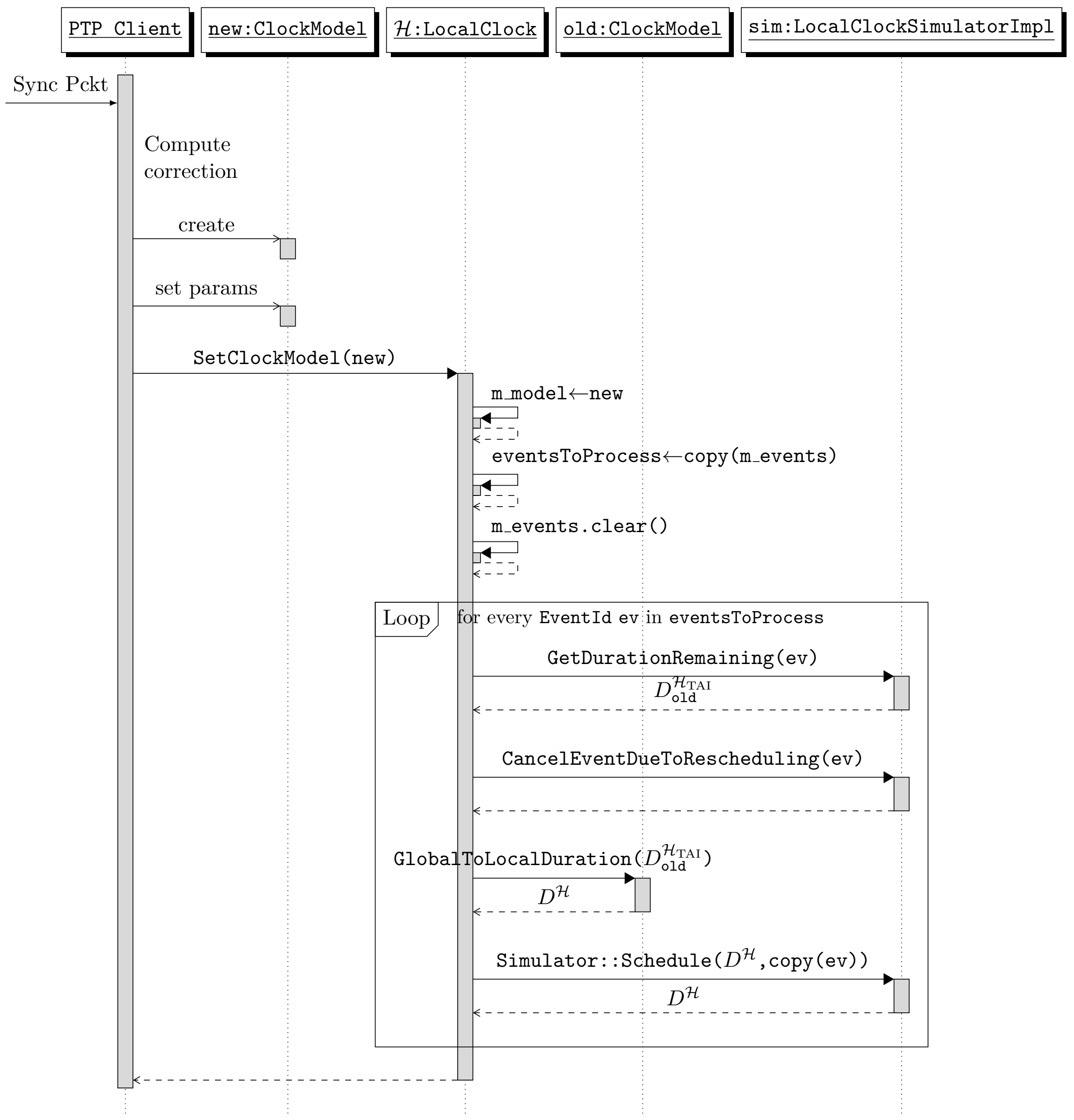

PTP Client
new:ClockModel
$\mathcal{H}$:LocalClock
old:ClockModel
sim:LocalClockSimulatorImpl
Sync Pckt
Compute correction
create
set params
SetClockModel(new)
m_model←new
eventsToProcess←copy(m_events)
m_events.clear()
Loop
for every EventId ev in eventsToProcess
GetDurationRemaining(ev)
$D^{\mathcal{H}_{\text{TAI}}}_{\text{old}}$
CancelEventDueToRescheduling(ev)
GlobalToLocalDuration($D^{\mathcal{H}_{\text{TAI}}}_{\text{old}}$)
$D^{\mathcal{H}}$
Simulator::Schedule($D^{\mathcal{H}}$,copy(ev))
$D^{\mathcal{H}}$

Figure 7.7: Sequence diagram of the proposed design for the use-case "Updating the Clock Model". The clock model is a private attribute of the `LocalClock`, therefore the only way for external classes to update the clock model is to call `Local::SetClockModel`. This gives an opportunity to the `LocalClock` for identifying all the events `ev` that have been scheduled in the $\mathcal{H}_{\text{TAI}}$-aligned `Scheduler` using the previous clock model, compute their remaining time in $\mathcal{H}_{\text{TAI}}$ with the new clock model and reschedule them.

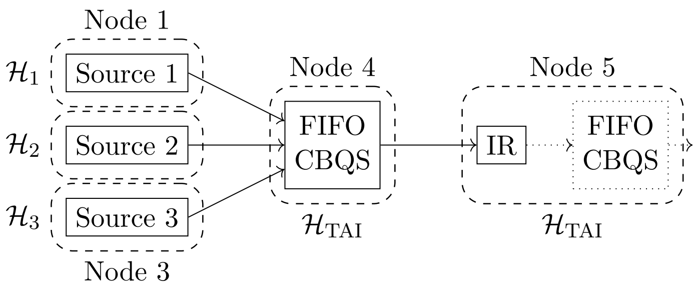


Figure 7.8: Scenario showing the instability of the non-adapted IR in synchronized networks.

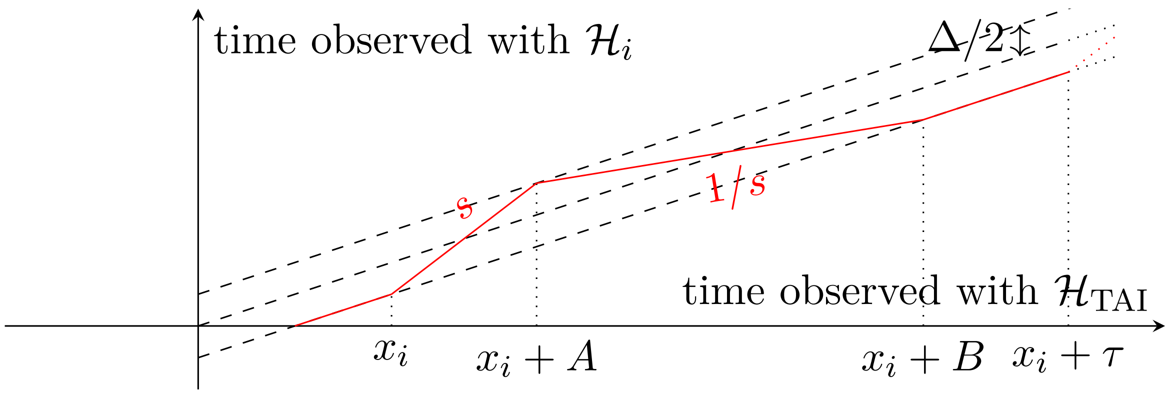


Figure 7.9: Shape of the function $h_i(t)$ for the adversarial clock $\mathcal{H}_i$.

7.4 Application: Simulating the IR Instability with Non-Ideal Clocks

To prove the instability of the non-adapted IR in a synchronized network with non-ideal clocks, the proof of Proposition 5.10 uses the scenario presented in Figure 7.8. Three nodes (on the far-left) have each a local clock $\mathcal{H}_i$ and a source application that generates a unique flow. The three flows have the same leaky-bucket arrival curve at their source $\gamma_{r,b}$ (when observed with the source's clock $\mathcal{H}_i$). The three flows then reach a fourth node with an ideal clock ($\mathcal{H}_{\text{TAI}}$) and they compete to exit Node 4 through one of its output ports made of a FIFO CBQS with no optional function. Last, the aggregate made of the three flows reaches a fifth node, with an ideal clock $\mathcal{H}_{\text{TAI}}$ and a non-adapted IR (it enforces $\gamma_{r,b}$ for each flow).

We denote by ρ, η and Δ the parameters of the synchronized time model (clock-stability bound, time-jitter bound, synchronization precision, Chapter 5) for the synchronized network made of Nodes 1 to 5. Proposition 5.10 states that, for any values of $\rho > 1$, $\eta \geq 0$ and $\Delta > 0$, there exists an adversarial beahavior of the three clocks $\mathcal{H}_1$, $\mathcal{H}_2$, $\mathcal{H}_3$ that respects the boundaries of the time model and an adversarial generation of the three flows that respects the $\gamma_{r,b}$ arrival-curve constraints such that the delay (observed with $\mathcal{H}_{\text{TAI}}$) of the packets in the IR diverges.

To do, so, the proof selects, for each clock $\mathcal{H}_i$, the behavior illustrated by the time function h_i in Figure 7.9: When the true time $\mathcal{H}_{\text{TAI}}$ reaches x_i, the clock $\mathcal{H}_i$ increase its frequency, set to $s > 1$. Then at $x_i + A$, it slows down and its frequency is set to $\frac{1}{s} < 1$ until the true

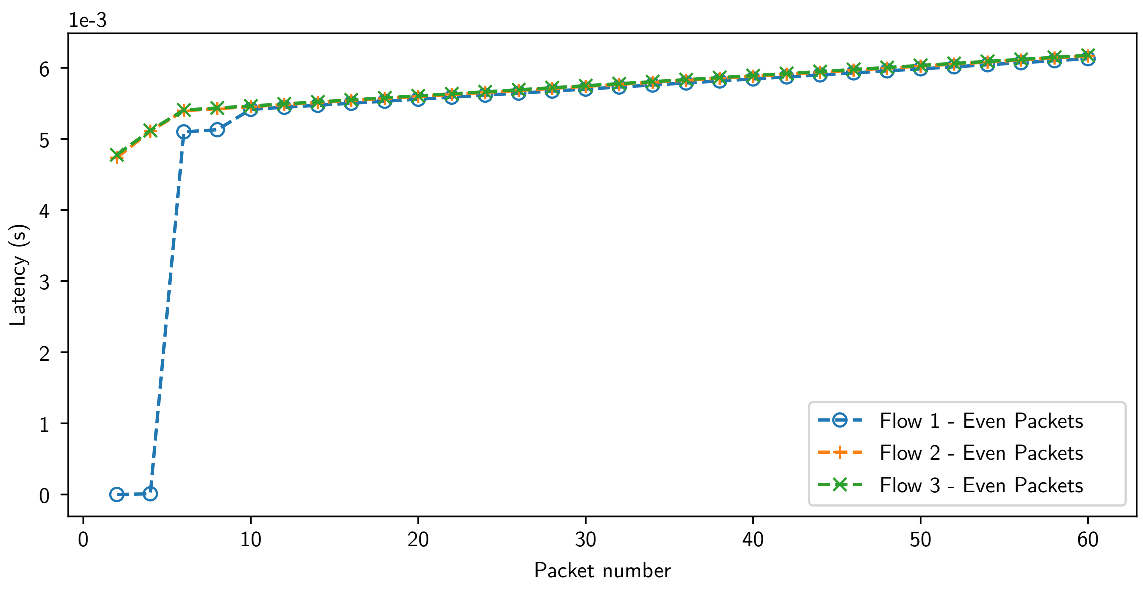


Figure 7.10: Delay spent in the IR as a function of the packet number. Result of the ns-3 simulation of the adversarial trajectory when $\rho = 1.002$, $\eta = 0$, $\Delta = 1\mu$s, $r = 149.8$kB/s and $b = 1498$B.

time reaches $x_i + B$. At $x_i + \tau$, it repeats the same profile. The values A, B and the period τ do not depend on i and are selected by the adversarial model depending on the values for ρ, η, Δ, r and b. All three clocks $\mathcal{H}_1$, $\mathcal{H}_2$ and $\mathcal{H}_3$ adopt the same profile, but they do not start at the same time; x_i depends on i and is again selected by the adversarial model.

To model these clocks on ns-3, Guillermo created a sub-class `AdversarialClock` of `ClockModel` [6][11]. He then combined all the pieces of his master project [Aguirre Rodrigo 2020] to simulate the situation of Figure 7.8:

Figure 7.10 presents the delay spent by each packet inside the IR, as a function of its packet number and obtained through ns-3. It is obtained for the case $\rho = 1.002$, $\eta = 0$, $\Delta = 1\mu$s, $r = 149.8$kB/s and $b = 1498$B. To ease the reading of the figure, we show only the delay spent by the packets that have an even packet number[12]. After some time, we observe that the delay increases linearly with the packet number, which highlights the delay divergence. In this simulation, the delay that the packets spend in the IR increases by 934μs every second.

Conclusion

In this chapter we have described a proposed modification of the ns-3 source code for enabling the simulation of local clocks. In Section 7.1, we have introduced the key concepts of the ns-3 network simulator, some of which have inspired the design of xTFA (Chapter 6). Then

[11]Another option was to split the model into three parts ($[x_i, x_i + A]$, $[x_i + A, x_i + B]$, $[x_i + B, x_i + \tau]$), use an affine `PerfectClockImpl` to model each part, an trigger the 'swapping' between the various parts by using `LocalClock::SetClockModel()`.

[12]They correspond to the packets of type A^2 in the proof, Appendix B.3.10.

we have discussed the related work in Section 7.2 and have identified the limitations of the previous attempts. From these limitations we have derived a new set of requirements.

In Section 7.3, we have described the proposed solution through a class diagram and two use-cases. The solution was co-designed by Guillermo Aguirre Rodriguo and this thesis' author; it was implemented by Guillermo.

Finally in Section 7.4, we have discussed how Guillermo used the solution described in this chapter to simulate a non-adapted interleaved regulator (TSN ATS) in a tightly synchronized network [Aguirre Rodrigo 2020]. His work enables the validation of Proposition 5.10 through a simulation: The ns-3 simulations prove that the interleaved regulator (IR) and its implementation within TSN, *asynchronous traffic shaping*, can yield unbounded latencies if placed in a network of non-ideal clocks (whatever the synchronization precision $\Delta > 0$) and if its parameters are not adapted to take into consideration this clock non-idealities.

As the implementation of local clocks is of interest beyond the simulation of the IR and even beyond the simulation of TSN components, we have submitted the local-clock module to the ns-3 community, through a merge request on the mainline code [1]. The ns-3 maintainers have expressed their interest in the module through their positive feedback, both on the ns-3 mailing list [3; 7] and on the merge request [1].

The design detailed in this chapter is the one submitted originally. The ns-3 maintainers have identified a set of modifications on their side (on the `core` module) that could ease our implementation and reduce the need for workarounds [5]. It is expected that once these modifications have been performed on their side and once our implementation is adapted to these new modifications, the request could be merged on the ns-3 mainline code.

This chapter constitutes the second major practical contribution developed/co-developed during the thesis. In the next chapter, we conclude our practical contributions by highlighting the applicability of our theoretical contributions and the utility of xTFA on an industrial use-case.

Chapter 8

Application of the Contributions to an Industrial Use-Case

Si je vous ai raconté ces détails sur l'astéroïde B 612 et si je vous ai confié son numéro, c'est à cause des grandes personnes. Les grandes personnes aiment les chiffres. Quand vous leur parlez d'un nouvel ami, elles ne vous questionnent jamais sur l'essentiel. Elles ne vous disent jamais: "Quel est le son de sa voix ? Quels sont les jeux qu'il préfère ? Est-ce qu'il collectionne les papillons ?" Elles vous demandent: "Quel âge a-t-il ? Combien a-t-il de frères ? Combien pèse-t-il ? Combien gagne son père ?" Alors seulement elles croient le connaître.

Antoine de Saint-Exupéry, *Le Petit Prince.*

Contents

8.1 Description of the Industrial Use-Case 153

8.2 Side-Effects of the Multi-Path Topology on Latency Bounds: Contributions of Chapter 3 154

8.3 Side-Effects of the Redundancy Mechanisms on Latency Bounds: Contributions of Chapters 4 and 6 156

8.4 Side-Effects of the Synchronization on Latency Bounds: Contributions of Chapters 5 and 6 157

In this chapter, we analyze the worst-case latency bounds on an industrial use-case and highlight the applicability of our contributions. We first describe the industrial network in Section 8.1. In Section 8.2, we discuss the contributions of Chapter 3, the partial deployment of regulators using LCAN and the benefit of FP-TFA. In Section 8.3, we discuss the contributions of Chapter 4, the tight model of the packet-elimination function (PEF) and the use of regulators after the PEF. Finally in Section 8.4, we introduce clock non-idealities and we discuss the contributions of Chapter 5 and the regulator adaptation methods.

8.1 Description of the Industrial Use-Case

We consider the Volvo core TSN network described in [Navet, Bengtsson, Migge 2020].

Physical Topology: The physical topology is shown in Figure 8.1 and is based on [Navet, Bengtsson, Migge 2020, p. 4]. The network contains two redundant vehicule-level control units `P1` and `P2` [Navet, Bengtsson, Migge 2020, Page 4]. They run safety-critical functions. The network also contains four sub-system-level main control units (MCUs) that act as gateways between local networks (based on CAN) and the core TSN network.

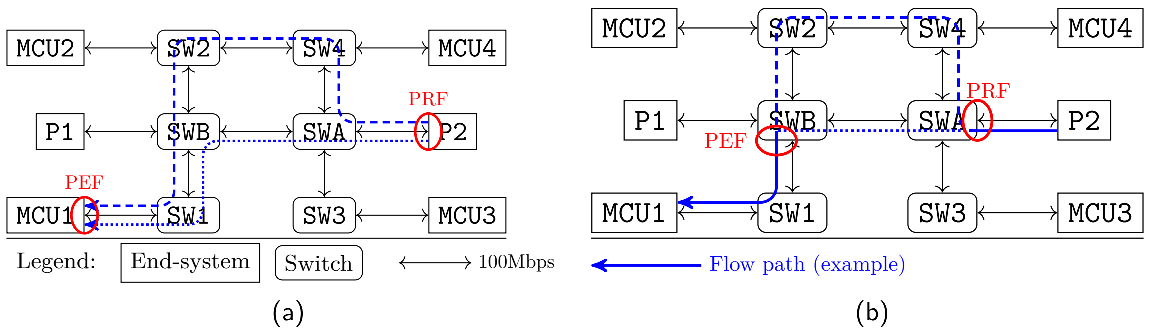


Figure 8.1: Simplified physical topology of the Volvo core TSN Network. From [Navet, Bengtsson, Migge 2020]. (a) Illustration of a flow in the configuration with redundancy mechanisms at the end systems. (b) Illustration of a flow in the configuration with redundancy mechanisms within the switches.

Table 8.1: Description of the Flows in the Use-Case. (a) Traffic profiles. (b) Flow path for $i \in \{1,2,3,4\}$, $p \in \{\mathtt{S},\mathtt{M1},\mathtt{M2},\mathtt{B}\}$

(a)

Name p	Payload size	Period at source
S	64B	81μs
M1	92B	324μs
M2	121B	567μs
B	150B	810μs

(b)

Flow name	Source	Dest.	Redundancy
C_MCUi_P12_p	MCUi	P1, P2	For C_MCU3_P12_p [resp., C_MCU4_P12_p], dest. P2 [resp., P1] is not redounded
C_P1_MCUi_p	P1	MCUi	Except for C_P1_MCU1_p
C_P2_MCUi_p	P2	MCUi	Except for C_P2_MCU3_p

Flow Description: We focus on the *Command and Control* class and consider four different periodic traffic profiles within the class. Their characteristics are listed in Table 8.1a. For each traffic profile and for each MCU, there exists a multicast flow that carries the sensor data from the MCU to both P1 and P2 and a unicast flow per control unit (2 in total) that carries the commands from the control unit to the MCU (see Table 8.1b). In total, the network contains 48 flows, 16 of which are multicast, for a total of 64 pairs [flow, destination].

Service Description: Each CBQS offers to the aggregate a rate-latency service curve with a rate of 100Mbps and a latency of 2μs.

8.2 Side-Effects of the Multi-Path Topology on Latency Bounds: Contributions of Chapter 3

The topology of Figure 8.1 is a multi-path topology that provides alternatives routes. We can exploit this property to redound all the flows for which we can find two different [source, destination] paths (last column of Table 8.1b).

We first consider a configuration in which the redundancy functions are performed by the end systems (Figure 8.1a): Each end system sends two packets per each data unit. They take opposite directions when they reach the backbone ring (clockwise and counter-clockwise). The destination end system then eliminates the duplicates. This configuration corresponds to the second row of Table 4.2 in Chapter 4. As we have discussed in Chapter 4, analyzing this situation does not require the results of Chapter 4: the end-to-end latency bound is

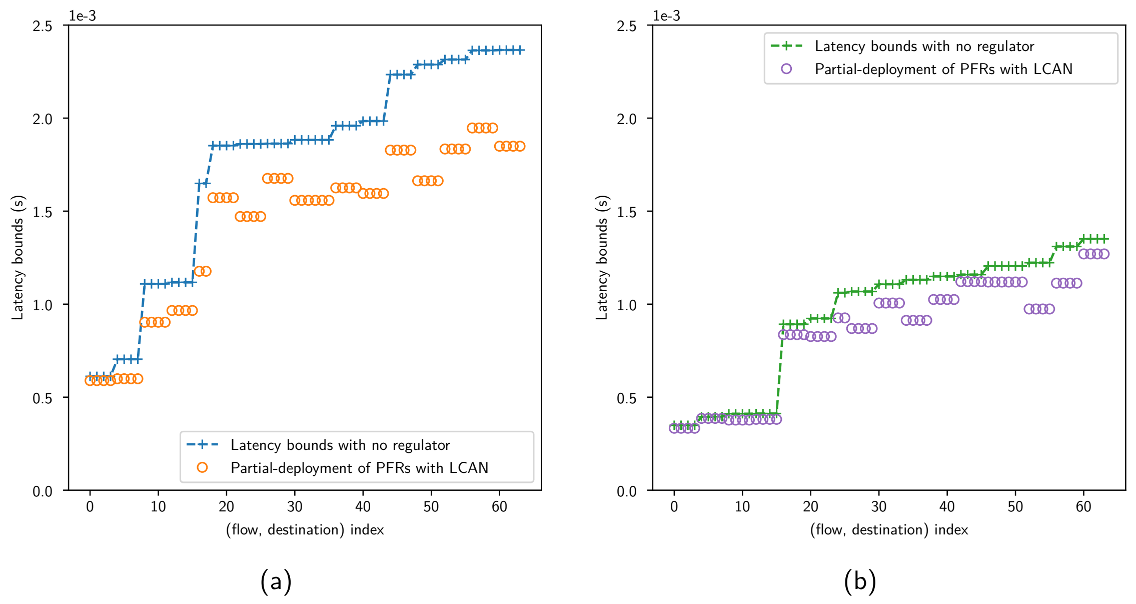


Figure 8.2: End-to-end latency bounds on the use-case. Comparison of the end-to-end latency bounds obtained with no regulators and with two PFRs placed as per the suggestion of the LCAN algorithm. (a) Latency bounds obtained when the redundancy functions are performed by the end systems. (b) Latency bounds obtained when the redundancy functions are performed by the switches of the backbone.

the maximum of the sub-flows latency-bounds. However, routing the sub-flows creates cyclic dependencies that are a side-effect of multi-path topologies. They are analyzed in Chapter 3.

The FP-TFA algorithm (Chapter 3, Section 3.3) can compute end-to-end latency bounds despite the presence of cyclic dependencies and even without traffic regulators for breaking them (*no-deployment* approach). We obtain the latency upper-bounds marked with a blue line in Figure 8.2a.

We now consider our *partial-deployment* approach of the traffic regulators for breaking the cyclic dependencies. The network contains two cyclic dependencies (clockwise and counter-clockwise). Our LCAN algorithm (Chapter 3, Section 3.2) finds the optimal solution with either two PFRs of two IRs. For the PFRs, the suggested positions are at output port $\mathtt{SWB}_{\mathtt{North}}$ to process the flows coming from `SWA` and at $\mathtt{SWB}_{\mathtt{East}}$ to process the flows from `SW2`.

The end-to-end latency bounds obtained with this *partial-deployment* approach are marked with orange circles in Figure 8.2a. We note that the *partial-deployment* approach provides a noticeable improvement with respect to the *no-deployment* approach, especially for the flows with a high end-to-end latency bound.

We can explain this important gain of the *partial-deployment* approach by computing the network load for the use case: we obtain a load greater than 98%. This important load worsens the burst-propagation effect on the backbone ring, which worsens in turn the effect of the cyclic dependencies on the latency bounds. Only one regulator per direction is required to break the cyclic dependencies, hence avoiding the burstiness to propagate, which improves the latency bounds in particular for the burstiest flows (with the highest latency bounds).

8.3 Side-Effects of the Redundancy Mechanisms on Latency Bounds: Contributions of Chapters 4 and 6

We now consider a scheme in which the redundancy functions are performed by the backbone switches (Figure 8.1b): Each end system sends only one packet per each data unit. When a data unit reaches the backbone ring, the switch sends one replicate in each direction, then the remote switch eliminates the duplicates before sending the resulting flow to the edge link towards the destination end system. This corresponds to the last row of Table 4.2 in Chapter 4.

As discussed in Chapter 4, the literature prior to this thesis does not enable the computation of performance bounds with network calculus for this configuration. Latency bounds can here be computed based on our toolbox of results in Chapter 4, Section 4.3 and their implementation in xTFA (Chapter 6).

The end-to-end latency bounds obtained by xTFA for this configuration are shown in Figure 8.2b. We first note that the obtained latency bounds are significantly better than those of Figure 8.2b, with a gain of 40%. Indeed, when the redundancy functions (duplication and elimination) are delegated to the backbone switches, the edge links between the switches and the end systems are less loaded (the network load is reduced to 85%), which reduces the burstiness and latency bounds for the output port connected to these links.

As in the previous configuration, we observe that xTFA computes end-to-end latency bounds even if the cyclic dependencies are not broken by traffic regulators. This is due to the extended fixed-point theorem for networks that contain PREFs (Theorem 4.3 in Chapter 4). However, using a partial deployment of traffic regulators to break the two cyclic dependencies (by relying again on the recommendation of LCAN) improves the latency bounds here as well. The gain is less important because the network load is smaller than in Figure 8.2a.

In Figure 8.2b, the traffic regulators are used for breaking the cyclic dependencies in the backbone ring. But as we have discussed in Chapter 4, traffic regulators can also be of interest when placed after the packet-elimination functions (PEFs) to remove the burstiness increase caused by the redundancy when the flows exit the backbone ring and compete to access the edge links towards their destination. Consider for example the four redounded flows from `P2` to `MCU1`, their path is shown in Figure 8.1b. Each of them is processed by a PEF within `SWB` to eliminate the duplicates coming from `SW2` and `SWA`. We evaluate the opportunity to shape the four flows after the PEFs and before they compete with the four other flows coming from `P1` in the output port of `SWB` towards `SW1`. We can either use four PFRs, or we can use a unique IR, because the four flows share the same reference point `P2`.

The latency bounds obtained with the various configuration options for $\texttt{SWB}_{\texttt{South}}$ are shown in Figure 8.3a. We focus here on eight flows: four non-redounded flows that come from `P1` towards `MCU1` and four redounded flows from `P2` towards `MCU1` as well. The baseline in green corresponds to the situation with only the PEF. We observe that as soon as we place a regulator (preceded or not by a POF), then the latency bounds of the non-redounded flows is improved because these four flows compete now with four redounded flows that exhibit a smaller burstiness. However, for the four flows that come from `P2` and are processed by the PEFs and the regulator, their latency bounds is improved only if the PFRs are preceded with per-flow POF and only if the IR is preceded by an aggregate POF. When the PFRs are

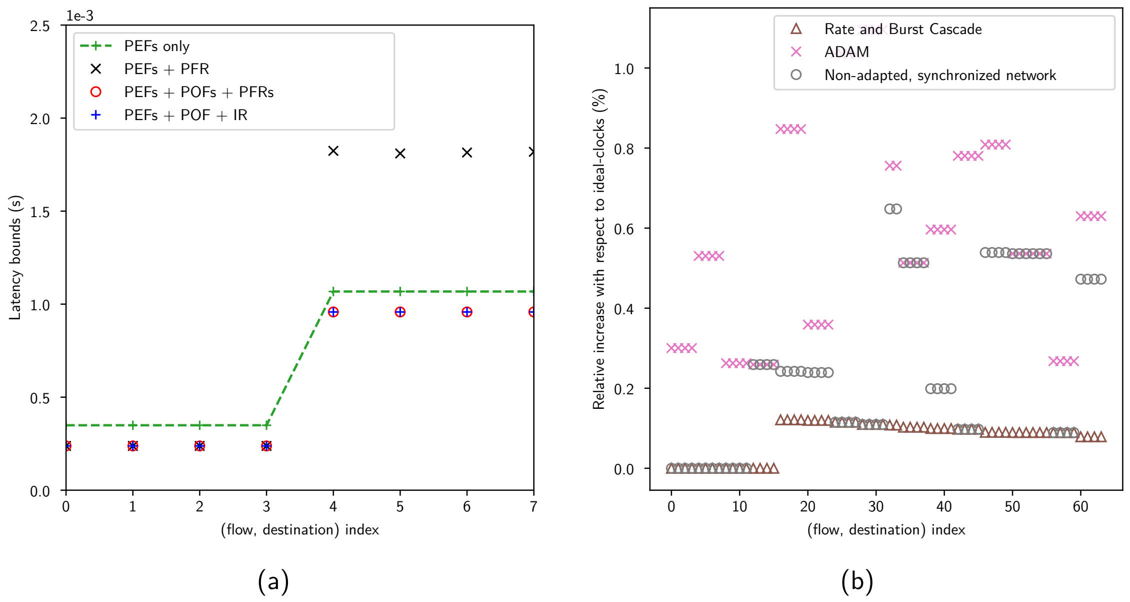


Figure 8.3: End-to-end latency bounds on the volvo use-case. (a) Comparison of the latency bounds obtained with different combinations inside the output port $\mathtt{SWB}_{\mathtt{South}}$. (b) Relative increase of the end-to-end latency bounds with respect to the ideal-clocks situation, with the different adaptation methods for adapting the regulators' configuration when the clock are non-ideal (η =4ns, $\rho = 1 + 2 \cdot 10^{-4}$ and, if synchronized, Δ =1μs).

not preceded by POFs the processed flows suffer a delay penalty as per Theorem 4.4. When the IR is not preceded by an aggregate POF, we cannot compute any latency bound as per Theorem 4.5.

8.4 Side-Effects of the Synchronization on Latency Bounds: Contributions of Chapters 5 and 6

Last, we evaluate the effect of clock non-idealities and of the synchronization mechanism on the latency bounds.

We consider again the situation of Figure 8.2b with two PFRs for breaking the cyclic dependencies in the backbone ring but no regulator placed after the PEFs on the edge links. We assume that the clocks are non-ideal. As the network contains two PFR, we know from Chapter 5, Proposition 5.5 that the PFRs cannot be left non-adapted if the network is non-synchronized. Otherwise they can yield unbounded latencies.

Section 5.5 of Chapter 5 provides three solutions: we can either keep a non-synchronized network and adapt the configurations of the two PFRs using the rate-and-burst cascade or the ADAM method; or we can synchronize the network.

We compute the end-to-end latency bounds of the flows for each option. Note that the clocks are no longer assumed to be ideal, hence we also use the toolbox of Chapter 5, Section 5.3, the "always in TAI" strategy (Section 5.4.1) and its implementation in xTFA (Chapter 6) to compute TAI delay bounds, even for the flows that are not processed by the two regulators.

The results are shown in Figure 8.3b. It represents the increase, in %, of the obtained TAI latency bounds with respect to the ideal-clock situation (that corresponds to the curve “partial deployment” of Figure 8.2b).

We observe that the rate-and-burst cascade provides the smallest latency bounds, which is consistent with our evaluation in Chapter 5, Section 5.6. The latency bounds computed with this adaptation method are less than 0.2% worse than in the ideal-clock situation. The ADAM adaptation method and the choice of using a synchronization mechanism provide similar end-to-end latency bounds. They all remain within 1% of the ideal-clock situation.

Figure 8.3b shows that, once the regulators are correctly configured (or if the network is synchronized and PFRs are used), then the penalty on the end-to-end TAI latencies is negligible with respect to the ideal-clock situation. However, it does not imply that the clock non-idealities can be neglected: clock non-idealities must be considered when configuring the regulators’ parameters (except for synchronized PFRs). If the network either is non-synchronized or contains IRs, then ignoring the clock non-idealities when configuring the regulators’ parameters can lead to unbounded latencies, as proved in Propositions 5.5, 5.10 and as simulated in Chapter 7.

Conclusion

In this section, we have analyzed the side effects of the new topologies (multi-path topologies) and of the new mechanisms (redundancy and synchronization mechanisms) on an industrial use-case based on [Navet, Bengtsson, Migge 2020]. The analysis has been conducted with our theoretical contributions of Chapters 3, 4, 5 and their implementation in the xTFA tool (Chapter 6).

We have first observed that routing redundant flows (with redundancy functions at the end systems) generate cyclic dependencies within the network’s backbone. FP-TFA is able to compute latency bounds despite the cyclic dependencies, but placing two PFRs at the locations suggested by LCAN improves the latency bounds, especially when the load of the network is high.

We have observed that the implementation of the redundancy functions at the switches (instead of the end systems) reduces the network load and the end-to-end latency bounds. Latency bounds are computed based on our framework for modeling PREFs within network calculus, on the extended fixed-point result and on xTFA. Here again, placing only two PFR breaks the cyclic dependencies and improves the end-to-end latency bounds of all flows.

We have then explored a second use of regulators: reshaping the flows that exit the backbone ring, after the elimination of their duplicates and before they compete with other flows to access the edge link towards their destination. We have observed that the non-processed flows benefit from placing such regulators because they compete with reshaped, less-bursty flows. However, the flows that are processed by the regulators benefit from the situation only if a packet-ordering function (POF) is placed after the PEF and before the regulator.

Last we have analyzed the effect of clock non-idealities on the latency bounds when the PFRs are used for breaking the cyclic dependencies. Latency bounds are computed based on our framework for modeling clock non-idealities and time synchronization within network calculus, on the proposed end-to-end strategies, on the adaptation methods, and on their implementation within xTFA. We concluded that if the clock non-idealities are not neglected when configuring the regulators, then their effect on latency bounds can be neglected. The synchronization mechanism has here a positive effect because no adaptation method is required for synchronized PFR; it provides latency bounds similar to the ADAM method.

In the next chapter, we provide our conclusive remarks and we summarize our contributions. We then discuss future research directions.

Part IV

Conclusion and Perspectives

Chapter 9

Conclusion and Perspectives

"Ah, well, nothing is permanent in this wicked world – not even our troubles."

Henri Verdoux

Charlie Chaplin, *Monsieur Verdoux*.

In this chapter, we summarize our contributions and discuss the research perspectives that have been opened through our work.

As identified in Chapter 1, the various Ethernet-based solutions for time-sensitive networks are expected to converge towards the mechanisms developed by IEEE TSN and IETF DetNet. These working groups provide a set of bounded-latency mechanisms that extend the traditional forwarding process of an Ethernet bridge for providing bounded latency.

The effects of these mechanisms and of their combinations on the performance guarantees of time-sensitive networks are studied in numerous occasions in the literature. Most of these analyses rely on the network-calculus framework (Chapter 2). The framework relies on an abstraction of the flows and network elements: the arrival curves and the service curves.

However, the TSN and DetNet working groups also defined new services that time-sensitive networks should provide, as well as new mechanisms and topologies for providing such services. For example, they encourage the use of multi-path topologies for minimizing the reconfiguration effort. They provide redundancy mechanisms for providing high reliability and a time-synchronization protocol for synchronizing the network's clocks.

Through a worst-case performance analysis of their combinations by using network calculus, we have proved that the new mechanisms and the new topologies have side-effects on the worst-case performance metrics of time-sensitive networks and on the latency bounds.

9.1 Summary of the Contributions

Our contributions are summarized in Table 9.1. For each set of new mechanisms and topologies identified in the introduction of the present thesis (Figure 1), we have provided several levels of contributions, as outlined in Table 9.1. We first provided a toolbox of results for modeling the new mechanisms (redundancy and synchronization) within the network-calculus theory. We then relied on these toolboxes to develop the approaches and algorithms for computing end-to-end latency bounds in networks that contain cyclic dependencies, redundancy mechanisms, and/or non-ideal clocks. We have studied separately the interactions between the new mechanisms and the traffic regulators, such as PFRs and IRs, because the latter do not have a known service-curve characterization, hence the nature of their interactions with the new mechanisms and topologies is not captured by the above toolboxes. When regulators can be seen as an opportunity (for example for breaking cyclic dependencies), we have

Table 9.1: Summary of The Contributions in This Thesis and Classification per Level.
The blue-filled cells represent the main theoretical contributions. The green-filled cell represents the main practical contribution.

Contribution level	New topologies	New Mechanisms	
	Multipath topologies	Redundancy mechanisms	Time-Synchronization
Network-calculus toolboxes		**Redundancy within network calculus:** Effects on arrival curves and reordering (§4.3).	**Non-ideal clocks within network calculus:** Time models for synchronized and non-synchronized networks (§5.2). Changing the clock for a delay, an arrival curve a service curve (§5.3)
Computing end-to-end latency bounds.	**FP-TFA: Latency bounds in networks with cyclic dependencies** Any fixed-point of the feed-forward application is a valid bound (Thm. 3.2). Extension to networks with redundancy mechanisms (Thm. 4.3)		“Always in TAI” and “always in local time” strategies (§5.4).
Interactions with traffic regulators (PFRs and IRs).	*Partial-deployment* approach with LCAN (§ 3.2).	**Instability of the interleaved regulator implemented by TSN ATS** – when placed immediately after a packet-elimination function (Thm. 4.5), – when placed non-adapted in a network with non-ideal, even synchronized, clocks (Prop 5.10)	
		Bounded penalty with PFR. Roerdering the aggregate after elimination and prior to shaping solves the issue	Bounded penalty with synchronized PFR. Rate-and-burst cascade and ADAM adaptation methods (§ 5.5.2).
Tools	**Experimental modular TFA (xTFA)** (Chapter 6)		
			Local-clock module for ns-3 (Chapter 7).
Comparisons on parametric topologies and use-case	Comparison of the {*no*,*partial*,*full*} deployments (§3.4).	Comparison between our tight model and an intuitive approach.	Comparison of the two strategies and the two adaptation methods.
	Industrial use-case (Chapter 8)		

provided an approach for exploiting this opportunity (the algorithm LCAN). When the regulators lead to delay penalties (for example, with non-ideal clocks), we have provided methods for addressing the issue (rate-and-burst cascade and ADAM). Finally, we have provided tools based on our theoretical contributions that are used to compare the approaches on synthetic cases and on an industrial use-case.

9.1.1 Main Theoretical Contributions

The main theoretical contributions of this thesis (in blue in Table 9.1) are

A framework for modeling the effects of redundancy mechanisms within network calculus

The framework contains a model for packet replication, elimination and ordering functions (PREOFs) (Section 4.2) and a toolbox of results that bound the effect of such functions on the arrival curves (Theorem 4.1) and on the reordering (Theorem 4.2).

A framework for modeling the effects of clock non-idealities within network calculus, in synchronized and non-synchronized networks.

The framework contains a time model for synchronized and non-synchronized networks (Section 5.2.1), as well as a toolbox of results for changing the clock that observes a delay (Proposition 5.1), an arrival curve (Proposition 5.3), and a service-curve (Proposition 5.4).

The instability results for the interleaved regulator (implemented by TSN ATS) when associated with redundancy mechanisms or when placed in a network with non-ideal clocks.

Theorem 4.5 proves that the interleaved regulator (IR) can yield unbounded latencies if placed immediately after a PEF. Proposition 5.10 proves that the non-adapted IR is unstable when placed in networks with non-ideal clocks, even if the clocks are tightly synchronized.

The FP-TFA algorithm for computing latency bounds in networks with cyclic dependencies.

FP-TFA is based on the total-flow analysis (TFA) approach and uses a fixed-point formulation for computing latency bounds in networks that contain cyclic dependencies. It also takes into account the line-shaping effect (Section 3.3). Theorem 3.2 proves the validity of any fixed-point found by FP-TFA. And Theorem 4.3 extends theses results to networks with redundancy mechanisms.

9.1.2 Main Practical Contribution

The main practical contribution of the thesis (in green in Table 9.1) is

The xTFA tool for computing latency bounds in networks with the new mechanisms

The experimental modular TFA (xTFA, Chapter 6) provides a set of original data-structures and algorithms for obtaining end-to-end latency bounds in networks that can contain cyclic dependencies, redundancy mechanisms, non-ideal clocks, and regulators.

9.2 Perspectives

Here, we discuss two research directions based on the contributions provided in this thesis: the implementation of our results in compositional approaches and the analysis of the behavior of the interleaved regulator.

9.2.1 Adapting the Compositional Approaches to Compute End-to-End Latencies with the New Mechanisms in Time-Sensitive Networks

Implementing the results of our toolboxes within compositional approach is important for computing end-to-end performance bounds of real networks that implement the redundancy mechanisms and that have real (hence non-ideal) clocks. With xTFA, we have selected TFA, for its modularity and simplicity, which paves only the beginning of this research direction.

Towards a Generalization of the TFA Fixed-Point Theorem for Networks with Cyclic Dependencies

For example, in Section 6.3 of Chapter 6, we have listed all the restrictions that apply to xTFA when computing latency bounds in networks with cyclic dependencies and PREFs. In particular, the only diamond ancestor selected for the application of Theorem 4.1 (arrival curve at the PEF output) is the flow's source. This is because the fixed-point result with PREFs (Theorem 4.3) is limited to the vector that contains only the input bursts and the delay bounds from the source. It would be interesting to look for a generalization of this theorem.

Based on our work in [Thomas, Le Boudec, Mifdaoui 2019], Plassart and Le Boudec provide theoretical foundations to the fixed-point approaches for FIFO-per-class networks (without PREFs) in [Plassart, Le Boudec 2021]. They prove that the selection of the cuts (see Section 3.3.3) does not influence the obtained backlog bounds.

For example, with their SyncTFA algorithm, each iteration on the virtual feed-forward network can be split into two steps: (1) a *computation step* during which all network elements are processed simultaneously based on the assumed burst and delay bounds for the flows at their input and, (2) a *propagation step* during which the new burst and delay bounds obtained from the previous step are propagated to the next node in the respective flow path to become the new assumed values for a new iteration [Plassart, Le Boudec 2021].

In the absence of PREFs, the *propagation* step is straightforward because the flow paths are trees. But if we want to extend the SyncTFA approach to networks with PREFs, we could work on the *propagation step* to express that, for example, a node with a PEF receives the sum of the bursts bounds coming from the sub-paths and the maximum of their delay bounds. Then, we could generalize this principle to tuples that contain several delay-bound values from several ancestors.

SFA, PMOO: The Question of the Service Curve

For our results for PREFs, we have focused mainly on the arrival curves, which makes their implementations easy in the context of TFA. On the contrary, SFA and PMOO are service-curve oriented approaches, hence modeling PREFs in these compositional approaches, even for feed-forward networks, would require additional results.

An important challenge to address is the definition of a service curve for a non-lossless system. In our arrival-curve-oriented approach, we have concluded that the losses within a system S do not affect the arrival curve at the output of S, when delay bounds for the non-lost packets are already known through S (Lemma B.2). But obviously if the system loses packets, then it affects its service curve, as defined in Definition 2.6. Scaling elements [Ciucu, Schmitt, Wang 2011] have been used to model non-lossless systems and they could be extended to model redundancy mechanisms.

Modeling clock non-idealities in SFA and PMOO is much simpler, as the effect on service curves is described in Proposition 5.4. These compositional approaches would typically use the "always in TAI" strategy (Figure 5.7) by converting all the service curves within $\mathcal{H}_{\text{TAI}}$ before applying the network-calculus concatenation result (Theorem 2.3).

Linear Programming Approaches: Defining PREFs and Clocks through Constraints

As we have discussed in Chapter 2, Section 2.5, linear programming approaches do not directly rely on the notions of arrival and service curves, rather on the underlying constraints.

Therefore, modeling PREFs and clock non-idealities in PLP [Bouillard 2022] cannot directly benefit from the toolboxes proposed in this thesis. However, the models and the proofs that we have provided in this thesis can help with the identification of the underlying constraints that could model PREFs and clock non-idealities within PLP. In particular, both the non-synchronized and the synchronized time models can be described with linear constraints (Figure 5.2).

9.2.2 The Behavior of the Interleaved Regulator

Since its proposition under the name *Urgency-Based Scheduler* in [Specht, Samii 2016], the interleaved regulator (IR) has raised several questions about its behavior. The instability results provided in this thesis could shed some light on these questions.

The Quest of a Service-Curve Characterization

A key consequence of the service-curve characterization of the PFR (Proposition 3.1) is that small variations to the ideal situation of the shaping-for-free property, such as clock non-idealities, lead to bounded consequences, as observed in this thesis.

On the contrary, the interleaved regulator (IR) has no known service-curve characterization. Even worse, it can yield unbounded latencies, as soon as the clocks are non-ideal (and even when the synchronization precision is as tight as desired). This suggests that the IR exhibits a non-linear behavior incompatible with the service-curve model. Proving formally the non-existence of such characterization constitutes, hence a perspective for our instability results.

The Influence of the Non-monotonic FIFO Behavior on the Design of the Instability Proofs

The proofs of Proposition 5.10 (instability of the non-adapted IR in the synchronized time model) exploits the FIFO property of the IR by selecting an adversarial model with adversarial clocks and adversarial sources whose behavior leads to unbounded latencies (Appendix B.3.10).

The interested reader can observe that the proof selects adversarial sources that are not greedy: The rate of the source varies periodically, between periods when its rate is close to the maximum allowed rate (the time elapsed between two packets is small) and periods when it sends at a much lower rate. This remark also applies to the sources used in the proof of Theorem 4.5 (instability of the IR after a PEF).

We highlight the similarity of this observation with an observation from [Andrews 2009]: When proving that the FIFO policy is not stable in non-feed-forward networks at arbitrarily low network load, Andrews cites the work of [Bramson 1996] and [Gamarnik 1998] to explain that greedy sources cannot be used in his proof. Assume that each flow f_i in Andrew's proof is constrained at its source by $\alpha_{f_i,\phi} = \gamma_{\rho_i,\sigma_i}$. If, "*except for the burstiness allowed by σ_i [,] the data is always injected [by the source of f_i] at a constant rate ρ_i, then FIFO is stable. However, if data can sometimes be injected [by the source of f_i] at a rate less than ρ_i, then FIFO can be unstable at arbitrarily small network loads.*" Hence, Andrews concludes that "*(somewhat counterintuitively) FIFO exhibits extreme non-monotonicity properties*" [Andrews 2009, §1].

Other similarities can be noted: In [Andrews 2009], the flows have cyclic paths dependencies and, in the proofs of Theorem 4.5 and Proposition 5.10, the flows have temporal cyclic dependencies with each other. Additionally, in the problem faced by Andrews, "Large bursts do not cause instability" [Hajek 2000]. And, in our proofs, the burstiness does not play a role in the instability[1].

It would be interesting to see to which extent this similarity holds. For example, does the IR remain stable in the situations studied in the current thesis if we assume minimum traffic constraints on the sources ?

[1]For Theorem 4.5; the burstiness of the flows plays a role, only when the path delay bounds are disjoints, *i.e.*, $[d_1, D_1]$, $[d_2, D_2]$ with $d_2 > D_1$. When the bounds intersect, then the burstiness does not play any role in the proof.

Cinquième partie

Résumé en Français du Contexte et des Contributions Principales

Contexte

Les réseaux temps-réel sont utilisés depuis les années 90 dans les systèmes *cyber-physiques* (e.g., voitures, avions, usines) pour interconnecter les senseurs, unités de contrôle et actuateurs et pour supporter les applications critiques telles que les boucles des contrôle. Avant 2010, les réseaux temps-réel n'étaient présents que dans quelques secteurs industriels (principalement les industries aérospatiale et automobile) et plusieurs technologies de réseaux temps-réel existaient pour les différents secteurs.

Contrairement à la qualité de service *au mieux* offerte par les réseaux de communication non-critiques (comme Internet), les réseaux temps-réels servent les flux de données avec un service *déterministe* qui contient des garanties sur les bornes de latence et une garantie d'absence de pertes par congestion. Ce service "central" est le focus principal de cette thèse. Plusieurs mécanismes, comme les *ordonnanceurs*, les *lisseurs* et la *préemption de frame* ont été développés pour fournir un tel service.

Il est primordial d'obtenir des bornes de latences et de *backlog* prouvées pour valider les contraintes de temps et l'absence de pertes par congestion lorsque de tels mécanismes sont utilisés. Plusieurs méthodes déterministes ont été développées pour calculer de telles bornes et sont largement utilisées dans la littérature pour calculer les effets des mécanismes ci-dessus sur les bornes de latences.

Cette thèse répond à deux principales tendances :

• Premièrement, il existe un besoin croissant de standardisation des réseaux temps-réel. Beaucoup d'industriels ont rejoint le groupe de travail *time-sensitive networking* (TSN, réseaux temps-réel) de l'*Institute of Electrical and Electronics Engineers* (IEEE, institut des ingénieurs en systèmes électriques et électronique) ou *deterministic networking* (DetNet, réseaux déterministes) de l'*Internet Engineering Task Force* (IETF, groupe de travail pour l'ingénierie d'Internet). Chaque groupe spécifie un ensemble de technologies qui sont indépendantes du secteur industriel. Les secteurs industriels peuvent ensuite sélectionner les technologies en fonction de leurs besoins.

• Deuxièmement, les réseaux temps-réel doivent fournir des services plus larges. Par exemple, les réseaux de l'IEEE TSN doivent fournir non seulement une *latence bornée*, mais aussi un haut niveau de *fiabilité*, un service de *synchronisation du temps* et une reconfiguration simplifiée avec l'utilisation de topologies à plusieurs chemins qui offrent des routes alternatives. De nouvelles topologies (topologies à plusieurs chemins) et de nouveaux mécanismes (redondance et synchronisation) ont été développés pour permettre ces services additionnels. Leur capacité à fournir lesdits services est étudiée dans la littérature.

Toutefois, la littérature n'a que peu étudié les effets de ces nouveaux mécanismes ou de leur combinaison sur le service de *latence bornée*. De plus les méthodes d'analyse déterministe existantes ne considèrent pas ces nouvelles topologies ou ces nouveaux mécanismes. Comme nous le démontrons dans cette thèse, ces méthodes doivent être adaptées en présence des nouveaux mécanismes et topologies, sinon elles peuvent obtenir des bornes de performance invalides.

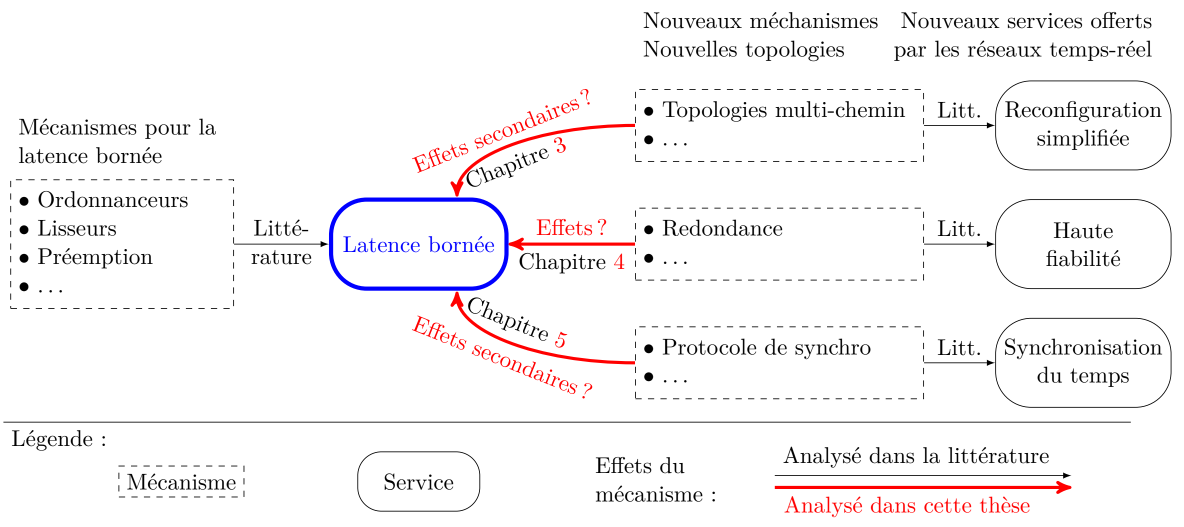


Figure 2 : Illustration du contexte de la thèse et de nos contributions théoriques. Le service de *latence bornée* dans l'ovale bleu est le focus de cette thèse. Plusieurs *mécanismes pour la latence bornée* (boîte en tirets sur la gauche) ont été développé pour fournir ce service et ont été étudiés à de nombreuses occasions dans la littérature. Un nouvel ensemble de services a été introduit dans le réseaux temps-réels (ovales sur la droite), pour lesquels de nouveaux mécanismes ont été développé. Leurs performances pour leur objectif respectif a aussi été étudiée dans la littérature. Dans cette thèse, nous étudions si ces nouveaux mécanismes et si ces nouvelles topologies peuvent avoir des effets sur le service de *latence bornée*. Figure basée sur [Farkas 2018, p. 5].

Dans cette thèse, nous analysons avec le calcul réseau les effets des combinaisons des nouveaux mécanismes (redondance, synchronisation du temps) et des nouvelles topologies sur les bornes de latences. Nous fournissons les fondations théoriques pour modeler les effets des nouveaux mécanismes dans la théorie du calcul réseau. Nous analysons aussi les interactions avec les régulateurs de trafic, en particulier le régulateur entrelaçé (*interleaved regulator*, IR, implémenté par IEEE TSN *asynchronous traffic shaping*). Nous montrons que les interactions entre les mécanismes de redondances, de synchronisation du temps et les régulateurs entrelacés peuvent mener à des latences non bornées.

Le manuscrit est organisé comme suit : Dans la première partie, nous introduisons le contexte et discutons la littérature. Dans le Chapitre 1, nous discutons le contexte technologique sur les réseaux temps-réels et l'évaluation de leurs performances. Dans le Chapitre 2, nous nous focalisons sur la théorie du calcul réseau. Nos contributions théoriques sont ensuite présentées dans la seconde partie : Dans le Chapitre 3, nous analysons les effets des dépendances cycliques sur les bornes de latences et leurs interactions avec les régulateurs de trafic. Dans le Chapitre 4, nous analysons les effets des mécanismes de redondance sur les bornes de latence et leurs interactions avec les régulateurs de trafic. Dans le Chapitre 5, nous analysons les effets d'une synchronisation imparfaite (ou de son absence) sur les bornes de latence et ses conséquences pour les régulateurs de trafic. Nos contributions pratiques sont présentées dans la dernière partie : Dans le Chapitre 6, nous détaillons l'outil xTFA (*experimental modular TFA*, TFA expérimental et modulaire), un outil pour calculer des bornes de latence dans les réseaux temps-réels. Cet outil implémente les résultats théoriques de cette thèse. Dans le Chapitre 7, nous détaillons un ajout au simulateur de réseaux ns-3. Cet ajout permet de simuler les imperfections des horloges et les problèmes que cela pose avec les IRs. Dans le Chapitre 8, nous présentons une application de nos résultats sur une étude de cas industrielle. Nous donnons nos conclusions et de futurs axes de recherche dans le Chapitre 9.

L'annexe A est conçue comme un *vade mecum* pour le manuscrit. Elle peut être imprimée séparément et contient une liste des acronymes, une table de notations et un glossaire. Dans toute la thèse, les termes soulignés sont définis dans le glossaire. L'annexe B contient les preuves des résultats théoriques.

Contributions Principales et Conclusion

Nous résumons ici nos contributions.

Comme identifié dans le Chapitre 1, les différentes solutions de réseaux temps-réel basées sur Ethernet devraient converger vers les mécanismes développés par IEEE TSN et IETF DetNet. Ces groupes de travail fournissent un ensemble de mécanismes qui améliorent le fonctionnement traditionnel d'un switch Ethernet afin d'offrir une latence bornée.

Les effets de ces mécanismes et de leurs combinaisons sur les garanties de performance des réseaux temps-réels sont étudiés à de multiples occasions dans la littérature. La plupart de ces analyses reposent sur la théorie du calcul réseau (Chapitre 2). Cette théorie repose sur une abstraction des flux et des éléments de réseau : les courbes d'arrivée et les courbes de service.

Cependant, les groupes de travail TSN et DetNet définissent également de nouveaux services que les réseaux temps-réels doivent fournir, ainsi que des nouveaux mécanismes et topologies pour offrir ces nouveaux services. Par exemple, ils encouragent l'utilisation de topologies à plusieurs chemins pour minimiser l'effort de reconfiguration ; ils offrent des mécanismes de redondance pour assurer un haut niveau de fiabilité ; ils fournissent enfin un protocole de synchronisation du temps pour synchroniser les horloges du réseau.

Via une analyse en calcul réseau de leurs combinaisons, nous avons prouvé que ces nouveaux mécanismes et ces nouvelles topologies ont des effets secondaires sur les métriques de performance pire-cas des réseaux temps-réels et sur les bornes de latences.

Résumé des Contributions

Nos contributions sont résumées dans la Table 1. Pour chaque ensemble de mécanismes et de topologies identifié dans l'introduction (Figure 2), nous offrons plusieurs niveaux de contributions, comme souligné dans la Table 1.

Tout d'abord, nous fournissons un ensemble de résultats permettant de modéliser les nouveaux mécanismes (redondance et synchronisation) dans la théorie du calcul réseau. Nous nous basons ensuite sur ces résultats pour développer les approches et algorithmes qui permettent d'obtenir des bornes de latence de bout en bout dans les réseaux qui contiennent des dépendances cycliques, des mécanismes de redondance et/ou des horloges non-idéales. Nous étudions séparément les interactions entre les nouveaux mécanismes et les régulateurs de trafic comme les PFRs et les IRs, car ces derniers ne possèdent aucune représentation en courbe de service connue. Lorsque les régulateurs représentent une opportunité (par exemple pour casser les dépendances cycliques), nous fournissons une approche qui exploite cette opportunité (l'algorithme LCAN). Lorsqu'au contraire ils induisent une pénalité sur la latence (par exemple, en présence d'horloges non-idéales), nous fournissons les méthodes pour résoudre ce problème (cascade de rate et de burst et ADAM). Enfin, nous fournissons des outils basés sur nos contributions théoriques et qui sont utilisés pour comparer nos approches sur des cas synthétiques et sur un cas industriel.

TABLE 1 : Résumé des Contributions de la Thèse et Classification par Niveau.
Les cases en bleu représentent les principales contributions théoriques. La case en vert représente la principale contribution pratique.

	Nouvelles topologies	Nouveaux mécamismes	
Niveau de contribution	Topologies à plusieurs chemins	Mécanismes de redondance	Synchronisation du temps
Résultats de calcul réseau		**Redondance au sein du calcul réseau :** Effets sur les courbes d'arrivées et sur l'ordre des paquets (§4.3).	**Horloges non-idéales au sein du calcul réseau :** Modèles d'horloges pour les réseaux synchronisés ou non (§5.2). Changer l'horloge qui observe un délai, une courbe d'arrivée ou de service (§5.3)
Obtenir des latences de bout en bout.	**FP-TFA : Bornes de latences dans les réseaux avec dépendances cycliques** Tout point fixe de l'application *feed-forward* est une borne valide (Thm. 3.2). Extension aux réseaux avec des mécanismes de redondance (Thm. 4.3)		Stratégies "toujours dans le TAI" et "toujours dans le temps local" (§5.4).
Interactions avec les régulateurs de trafic (PFRs et IRs).	Approche du *déploiment partiel* avec LCAN (§ 3.2).	**Instabilité du régulateur entrelacé implémenté par TSN ATS** – placé immédiatement après une fonction d'élimination (PEF) (Thm. 4.5), – placé non-adapté dans un réseau avec des horloges non-idéales, même si elles sont synchronisées (Prop 5.10)	
		Pénalité bornée avec PFR. Ré-ordonner l'aggrégat après l'élimination et avant la régulation retire le problème.	Pénalité bornée avec un PFR synchronizé. Méthodes d'adaptation de contrat *cascade de rate et de burst* et ADAM (§ 5.5.2).
Outils	**TFA expérimental et modulaire (xTFA)** (Chapitre 6)		
			Module d'horloges locales pour ns-3 (Chapitre 7).
Conparaisons sur des topologies paramétriques et sur un cas d'étude	Comparaison des déploiments nul, partiel et total (§3.4).	Comparaison entre notre modèle précis et une approche intuitive.	Comparaison des deux stratégies et des deux méthodes d'adaptation.
	Étude de cas industrielle (Chapitre 8)		

9.2.3 Contributions Théoriques Principales

Les contributions théoriques principales de cette thèse sont (en bleu dans la Table 1) :

Une théorie pour modéliser les effets des mécanismes de redondance au sein du calcul réseau

Cette théorie contient un model pour les fonctions de réplication, d'élimination et d'ordonnancement des paquets ainsi qu'un ensemble de résultats pour borner l'effet de ces fonctions sur les courbes d'arrivée (Théorème 4.1) et sur le réordonnancement des paquets (Théorème 4.2).

Une théorie pour modéliser les effets des imperfections des horloges au sein du calcul réseau, dans les réseaux synchronisés ou non.

La théorie contient un modèle d'horloge pour les réseaux synchronisés ou non (Section 5.2.1), ainsi qu'un ensemble de résultats pour changer l'horloge qui observe un délai (Proposition 5.1), une courbe d'arrivée (Proposition 5.3) ou une courbe de service (Proposition 5.4).

Les résultats d'instabilité du régulateur entrelacé (implémenté par TSN ATS), lorsqu'il est associé à des mécanismes de redondance ou lorsqu'il est placé dans un réseau avec des horloges imparfaites.

Le théorème 4.5 prouve que IR peut mener à des latences non bornées lorsqu'il est placé immédiatement après une PEF. La proposition 5.10 prouve qu'un IR est instable lorsqu'il est placé non-adapté dans un réseau avec des horloges imparfaites, même lorsqu'elles sont synchronisées.

L'algorithme FP-TFA pour calculer des bornes de latences dans les réseaux avec des dépendance cycliques.

FP-TFA est basé sur l'approche total-flow analysis (TFA) et utilise une formulation de point fixe pour calculer des bornes de latences dans les réseaux qui contiennent des dépendance cycliques. Il prend aussi en compte le phénomène de lissage de ligne (line-shaping effect, Section 3.3). Le théorème 3.2 prouve la validité de tout point fixe trouvé par FP-TFA. De plus, le théorème 4.3 étend ces résultats à des réseaux qui contiennent aussi des mécanismes de redondance.

9.2.4 Contribution Pratique Principale

La contribution pratique principale de cette thèse (en vert dans la table 1) est :

L'outil xTFA pour calculer des bornes de latences dans les réseaux avec des nouveaux mécanimes

L'outil *TFA expérimental et modulaire* (xTFA, Chapitre 6) fournit un enemble de structures de données et d'algorithmes originaux pour obtenir des bornes de latence dans les réseaux qui peuvent contenir des dépendances cycliques, des horloges imparfaitess et des régulateurs de trafic.

Part VI

Appendices

Appendix A

Vade Mecum

This appendix can be printed separately. It contains a list of acronyms (§A.1), a glossary (§A.2), and a section that regroups the general notations as well as the reference model for a TSN bridge used throughout the manuscript (§A.3).

A.1 List of Acronyms

ADAM	asynchronous dual arrival-curve method
ACP	aggregate computation pipeline
AFDX	Avionics Full-dupleX switched Ethernet
API	application programming interface
ASIL	Automotive Safety Integrity Level
ATS	asynchronous traffic shaping
AVB	Audio Video Bridging
CAN	Controller Area Network
CBQS	class-based queing subsystem
CBS	credit-based shaper
DAG	directed acyclic graph
DetNet	deterministic networking
DES	discrete-event simulator
ETE	end-to-end
EP	elimination-pending
FIFO	first in, first out
FRER	frame replication and elimination for redundancy
GIF	graph induced by flows
GNSS	global navigation satellite system
GPS	global positioning system
gPTP	generalized PTP
GUI	graphical user interface
HSR	High-availability Seamless Redundancy
INCOSE	International Council on Systems Engineering
IEC	International Electrotechnical Committee
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IR	interleaved regulator
ISAE	*Institut Supérieur de l'Aéronautique et de l'Espace*
ISO	International Organization for Standardization
ITU	International Telecommunication Union
LAN	local area network

LCAN	low-cost acyclic network
LUDB	Least Upper Delay Bound
MCU	main control unit
MFAS	minimum feedback arc set
MFVS	minimum feedback vertex set
NC	network calculus
NTP	Network Time Protocol
PBOO	pay burst only once
PEF	packet-elimination function
PFR	per-flow regulator
PMOO	pay multiplexing only once
POF	packet-ordering function
PRF	packet-replication function
PREFs	packet replication and elimination functions
PREOFs	packet replication, elimination and ordering functions
PRP	Parallel Redundancy Protocol
PTP	Precision Time Protocol
QoS	quality of service
RBO	reordering byte offset
REG	regulator
RTO	reordering late time offset
SFA	separated flow analysis
TAI	international atomic time (*temps atomique international*)
TAS	time-aware shaper
TCP	Transmission Control Protocol
TDEV	time deviation
TFA	total-flow analysis
TG	task group
TSA	Transmission Selection Algorithm
TSN	time-sensitive networking
UML	Unified Modeling Language
VBR	variable-bit-rate
WG	working group
xTFA	experimental modular TFA

A.2 Glossary

aggregate An aggregate is a set of flows that is viewed as a unique flow. For example, for the aggregate $\{f_1, f_2\}$ and an observation point v, the cumulative arrival function of the aggregate at v is $R_{f_1,v} + R_{f_2,v}$ where $R_{f_i,v}$ is the cumulative arrival function of f_i at v. Unless specified otherwise, we often use 'aggregate' to describe the class aggregate, *i.e.*, the aggregate that contains all the flows of the class-of-interest . 24, 74

asynchronous A network/bus is asynchronous if the emission date of each data unit is unknown. The emission of the data unit is triggered by an external event, *i.e.*, the sources are *event-triggered*. An asynchronous network can be time synchronized or not . 8–10, 14, 21

best-effort A network offers a best-effort quality of service if it does not provide any guarantees on the performance metrics of the service that it offers to the flows . 6

causal A system is causal if it does not produce any data, does not duplicate any data, and does not expand the data passing through it . 20–24, 76, 77

clock-stability bound For a given set of clocks (by default, for the set of all the clocks in the network), the clock-stability bound is an upper-bound on the linear evolution of the relative time-function for any pair of two clocks in the set. It upper-bounds both the frequency offset and the low-frequency noise (the *wander*). It is denoted ρ . 99, 121, 125, 150, 188

curve A function of $\mathfrak{F}_0$. A function $\mathfrak{f} : \mathbb{R} \to \mathbb{R}^+ \cup \{+\infty\}$ is a curve if $\mathfrak{f}$ is wide-sense increasing and $\forall t \leq 0, \mathfrak{f}(t) = 0$. 18

data unit A piece of information sent from one sending application to one or several recipient applications. We always assume that a data unit is sufficiently small and does not need to be fragmented in order to be transported by the network. A data unit is an abstract concept that, in the context of PREOFs, can be located at several positions in the network at the same time, thus being transported by several packets at the same time (the *replicates*). A data unit can also cross an observation point several times, each time transported by a different packet . 5, 6, 31, 66, 68, 70, 80, 121, 154, 156

deterministic A network offers a deterministic service if it provides a guarantee on the maximum latency, maximum jitter, and maximum packet re-ordering for each flow. Optionally, it can also provide a time-synchronization service and a low packet loss-ratio, as in the contect of IEEE TSN . 6–8, 12, 13, 15

diamond ancestor For a, n two vertices of a flow graph $\mathcal{G}(f)$, a is a diamond ancestor of n in $\mathcal{G}(f)$ if a is not an EP-vertex of $\mathcal{G}(f)$ and all paths in $\mathcal{G}(f)$ from the graph root to n contain a. See Definition 4.3 . 71

duplicate At a given observation point [respectively, for a given function], a packet is a duplicate if another replicate of the same data unit previously crossed the observation point [respectively, previously entered the function] . 65, 68, 70–72, 156

feed-forward A feed-forward network is a network without any cyclic dependencies. In other words, its graph induced by flows (Definition 2.9) is acyclic. . 26, 38, 39, 134, 136

FIFO-per-class network A FIFO-per class network is a network in which each output port of each device offers one class-based queuing subsystem (CBQS) per class that is FIFO for the class. Note that the network itself is not necessarily end-to-end FIFO for the entire class; the adjective refers only to the individual output ports . 24, 26, 34

flow A coherent sequence of data units that originate at the same end system (the *source*) and that follow a path to reach one or several destinations . 6

greedy A source is greedy if it outputs the maximum amount of data that its contract allows. For a flow f, denote by $\alpha_{f,\phi}$ its specified source arrival-curve and by $R_{f,\phi}$ its cumulative arrival-function at the output of its source. The source of f is greedy if there exists a time instant $t_0 \geq 0$ such that, $\forall t \geq t_0, R_{f,\phi}(t) = \alpha_{f,\phi}(t - t_0)$. 94, 168

jitter For a flow or a network, the jitter is the variation of the latency of the data units of the flow when traversing the network. For a clock, the timing jitter is a high-frequency noise on the clock function h_i . 6

latency For a flow, a data unit or a network, the latency is the time needed by the flow or data unit to travel the network. For a service-curve element modeled by a rate-latency service curve, the latency is an upper-bound on the duration during which no service is guaranteed . 6

line-shaping effect The line-shaping effect describes the positive effect on the performance bounds of the serialization of the bits on the transmission links. This serialization is associated with a peak-rate limitation: Taking into account the line-shaping effect means computing better performance bounds due to this peak rate-limitation . 27, 29, 41, 44, 59–61, 88, 120, 128, 136, 165, 177

loss ratio For a flow, the number of data units that are lost in the network divided by the total number of data units sent by the flow's source . 6

lossless A network element is lossless if it does not lose any bit, any fraction of data . 20–24, 76, 167

packet A formatted unit of data that transports a data unit through the network. A packet is a virtual object that can be localized in the network using time and space . 31, 70, 154, 156

packetized A flow f is packetized at a given observation point w if for each packet p of f, the bits of the packet p cross w at the exact same time instant . 23, 24, 43, 78, 192, 215, 216

predictable A system or network has a predictable behavior if its state can be predicted at any time instant, based on the known properties of the network. A network with time-triggered sources and a global schedule has a predictable behavior: its state at any time instant can be derived from the global schedule . 7, 8

replicate The replicates of a data unit m are all the packets that transport m . 65, 70, 75, 156

scheduling policy The scheduling policy describes the algorithm or set of algorithms in a system that decide which packet to serve among those waiting for service. In a FIFO-per-class network, there exists a system-level and a class-level scheduling policies. The system-level scheduling policy decides which class to serve among those waiting. The class-level scheduling policy, that decides which packet to serve among those of the class waiting for service, is the FIFO policy . 24, 26, 32, 33

seamless redundancy A mechanism offers seamless redundancy if the time elapsed during the original emission of the data unit and the reception of the recovered data unit does not exceed the worst-case delay that is computed under normal operation. In other words, no extra delay is required to recover from the loss; the recovery time is zero . 64

sub-additive A function $\mathfrak{f} \in \mathfrak{F}$ is sub-additive if $\forall s, t \geq 0, \mathfrak{f}(s) + \mathfrak{f}(t)$. A sub-additive curve $\mathfrak{g}$ is a sub-additive function such that $\mathfrak{g}(0) = 0$. . 24, 43, 53, 215

synchronization precision For a given set of clocks (by default, for the set of all the clocks in the network), the synchronization precision upper-bounds the relative time difference between any pair of clocks. It is denoted Δ . 100, 121, 125, 150, 188

synchronous A network/bus is synchronous if the emission date of each data unit is known *a priori*, based on a global schedule. The sources are *time-triggered*: The emission of each data unit is triggered when the time reaches the programmed emission time of the data unit. A synchronous network must be synchronized (a common notion of time must be distributed to the node's constituent), but the synchronization can be limited to some nodes and its precision can vary depending on the properties of the synchronous access . 7, 9, 10, 14

system In the context of Chapter 1, a human-made system is a "*combination of interacting elements organized to achieve one ore more stated purposes*" [ISO/IEC/IEEE 15288, §4.1.46]. In all other chapters, a system is a delimited part of the network that is crossed by one or more flows. It has an input through which the data enters the system and an output through which the data exits the system. The system can be formed of a unique network element or by several network elements that do not need to be in series . 5, 20, 101, 108

time-jitter bound For a given set of clocks (by default, for the set of all the clocks in the network), the time-jitter bound is an upper-bound high-frequency *jitter* noise of the relative time-function for any pair of two clocks in the set. It is denoted η . 99, 121, 125, 150, 188

universally stable A scheduling policy is universally stable if any underloaded network of nodes that implement the policy is globally stable . 39

A.3 General Notations

Figure A.1 is a copy of Figure 2.10 in Chapter 2 and recalls the model of any TSN bridge, as used throughout the manuscript.

The general notations used thourough the manuscript are regrouped in Tables A.1 and A.2. Figure A.2 is a copy of Figure 2.12 in Chapter 2 and recalls the mapping between the graph induced by flows (GIF) and the devices as well as the locations of the observations points.

Table A.1: List of Notations

Notation	Description	See
	Network-calculus theory	
$\lvert c\rvert^+$	For $c \in \mathbb{R}$, $\lvert c\rvert^+ = \max(0, c)$	2.1
$\mathfrak{f}, \mathfrak{g}$	Functions $\mathbb{R} \to \mathbb{R} \cup \{+\infty\}$	|
$\mathfrak{F}$ [resp., $\mathfrak{F}_0$]	Set of wide-sense increasing functions $\mathfrak{f} : \mathbb{R} \to \mathbb{R} \cup \{+\infty\}$ such that $\forall t < 0$ [resp., $\forall t \leq 0$], $\mathfrak{f}(t) = 0$	|
$\gamma_{r,b}$	Leaky-bucket curve of rate r and burst b $\forall t > 0, \gamma_{r,b}(t) = rt + b$	|
γ_c	$= \gamma_{c,0}$	
$\beta_{R,T}$	Rate-latency curve with rate R and latency T $\forall t, \beta_{R,T}(t) = R\lvert t - T\rvert^+$	2.1
δ_D	Variable, D-bounded delay curve $\forall t \leq D, \delta_D(t) = 0 \quad \forall t > D, \delta_D(t) = +\infty$	|
	Flows, Graph induced by Flows (GIF) and Vertices	
f, g, h	Flows of the class-of-interest	
$\mathcal{F}$	A set of flows (an aggregate)	
$\mathcal{G}$	The graph induced by flows (GIF) of the network	2.6.6
$\mathcal{G}(f)$	The flow graph of f	|
n	A vertex in the GIF. The vertex comprises an output port and its remote input port.	|
$f \ni n$	Flow f crosses the vertex n.	|
$f \ni (p, n)$	Flow f crosses the vertex p and immedialty after the vertex n.	|
	Node model	
n	A vertex of the GIF, its output port or its CBQS	2.6
β_n	The service curve of the CBQS within n	2.6.4
c_n	The capacity of the transmission link within vertex n	2.6.5
T_n	The propagation time of the transmission link within n	|
$T_{\min}$	The minimum of T_n for all the vertices n of the GIF	|
	Observation points	
v, w	Observation points. They can be equal to:	2.6.7
ϕ	– the output of the flow's source	|
n^{in}	– the input of vertex n: the input of n's output port	|
n^*	– the output of vertex n: the output of n's remote in. port	|
n'	– within vertex n, at the output of n's output port (just before the serialization on the transmission link)	|
$n^\dagger$	– within vertex n, at the input of the CBQS in n's output- port (*i.e.*, at the output of the optional functions)	|
$\texttt{FUN}^{\text{in}}$	– the input of the optional function `FUN`	
$\texttt{FUN}^*$	– the output of the optional function `FUN`	

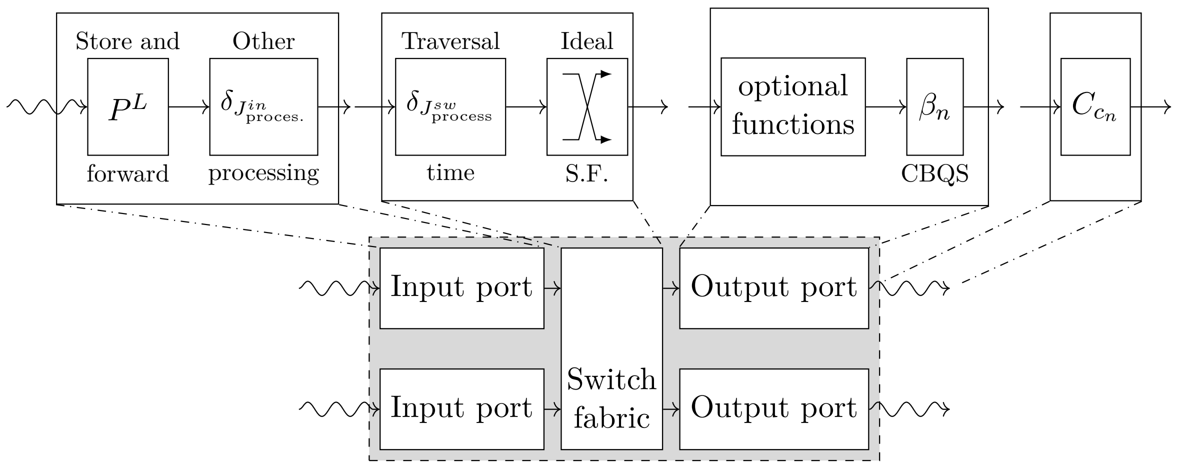


Figure A.1: Model for any device in the time-sensitive network, copy of Figure 2.10 in Chapter 2. Each device is made of input ports, output ports and a switching fabric. The input port contains a store-and-forward step and bounded technological latencies. The switching fabric contains an ideal switching fabric and bounded technological latencies. The output port contains a set of optional functions and a FIFO CBQS. The transmission links act as greedy shapers.

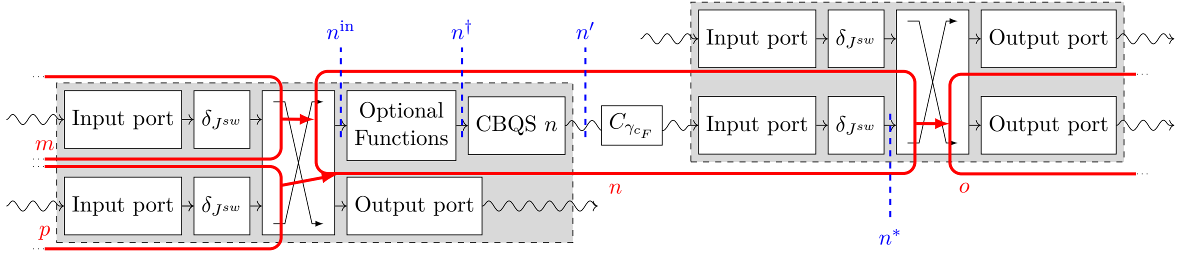


Figure A.2: Relation between the bridges (in gray boxes), the vertices of the underlying graph (in thick red ovals) and the observation points (dashed blue lines). The vertex n models the output port n together with the transmission link, the input port on the remote device and the latency of the switching fabric in the remote device. Because of the one-to-one mapping between output ports and vertices, the notation n describes equivalently the vertex n, the ouput port n or the CBQS n.

Table A.2: List of Notations (continued)

Notation	Description	See
	Properties of a flow at a given observation point	
$\alpha_{f,v}$	Arrival curve of f at the observation point v	2.2.3
r_f	Rate of the leaky-bucket-constrained flow f	
$b_{f,v}$	Burst of the leaky-bucket-constrained flow f at the observation point v	
$\lambda_{f,v}(w)$	Reordering late time offset (RTO) of f at v, with respect to its order at w	4.3.2
$\pi_{f,v}(w)$	Reordering byte offset (RBO) of f at v, with respect to its order at w	|
$z_{f,v}$	Flow state of f at v	6.2.2
	Delays	
$d_{f,S}$ [resp., $D_{f,S}$]	Lower [resp., upper] delay bound for f through S	
d_S [resp., D_S]	Lower [resp., upper] delay bound for the class aggregate through S	
$d_f^{v\to w}$ [resp., $D_f^{v\to w}$]	Lower [resp., upper] delay bound for f between observations point v and w	
	The optional functions: regulators, PEFs, POFs	
$\texttt{IR}_n(p)$	Interleaved regulator placed at n and processing the flows coming from p	3.2
$\texttt{PFR}_n(p)$	Set of parallel per-flow regulators placed at n and processing the flows coming from p	|
$\sigma_{f,n}$	Configured shaping curve for the flow f at the regulator processing f in vertex n	|
$\texttt{REG}_n(\mathcal{F}, w)$	Regulator (PFR if $\mathcal{F} = \{f\}$, IR otherwise) placed at n and that enforces for each flow $f \in \mathcal{F}$ the arrival curve that the flow had at w	4.2.2
$\texttt{POF}_n(\mathcal{F}, w)$	Packet-ordering function (POF) placed at n that enforces for the aggregate $\mathcal{F}$ the order that the packets had at w	|
$\texttt{PEF}_n(f)$	Packet-elimination function (PEF) placed at n that removes the duplicates from flow f	|
	Clocks and time measure	
t	The measure of a time instant	
T_{start}	When any of the clocks in the network shows T_{start}, all other clocks show positive values **and** no source has sent any bit yet.	5.2.1
$\mathcal{H}$	A clock	|
$\mathcal{H}_{\text{TAI}}$	The international atomic time (TAI)	|
η, ρ, Δ	The parameters of the non-synchronized and synchronized time models. Respectively: the time-jitter bound, the clock-stability bound and the synchronization precision	|
	Units	
t.u.	Arbitrary time unit used for the examples.	
d.u.	Arbitrary data unit used for the examples.	

Appendix B

Proofs

B.1 Proofs of Chapter 3

B.1.1 Proof of Theorem 3.1

Proof of Theorem 3.1. Consider a flow f. For any packet q of the flow, we denote by l_q its size, A_q the arrival of its first bit, B_q the arrival of its last bit, and D_q the release of the entire packet. By definition of the packetizer, when all the bits that belong to the packet have been received, the packet is released from the packetizer, we thus have $\forall q, D_q = B_q$. Furthermore, the transmission rate at the input is c. As the previous CBQS processes one packet at a time and does not interleave several packets, between A_q and B_q, only the bits of the packet q of flow f are received, thus $\forall q, B_q - A_q \leq \frac{l_q}{c}$. The packetizer is FIFO, thus all the bits belonging to packet q suffer a delay less than the delay suffered by the first bit $D_q - A_q = \frac{l_q}{c} \leq \frac{l_{\max}}{c}$

Thus, for any bit arriving at the packetizer, there exist a packet number q such that the bit belongs to Packet q and the delay of the bit through the packetizer is bounded by $\frac{l_{\max}}{c}$.

The packetizer is a causal, lossless and FIFO system in which the delay for any bit is bounded by $\frac{l_{\max}}{c}$. Applying [Le Boudec, Thiran 2001, Prop. 1.3.3] gives the result. □

B.1.2 Proof of Theorem 3.2

Proof of Theorem 3.2. $\mathcal{FF}$ can be assumed to be wide-sense increasing as per [Bouillard, Boyer, Le Corronc 2018, Chap. 12]. The iterations start with vector $\mathbf{0}$. Thus the obtained fixed-point $\overline{\boldsymbol{b}}$ is the lowest fixed-point of $\mathcal{FF}$ and it is then sufficient to prove that this vector is a valid bound for the bursts at the cuts.

Consider the view of the cyclic network in Figure B.1 where points U and W are located respectively after and before the cuts. The network between U and W is feed-forward.

Example: In Figure 3.12, $U = \texttt{S4}''_{\texttt{North}}$ and $V = \texttt{S4}'_{\texttt{North}}$.

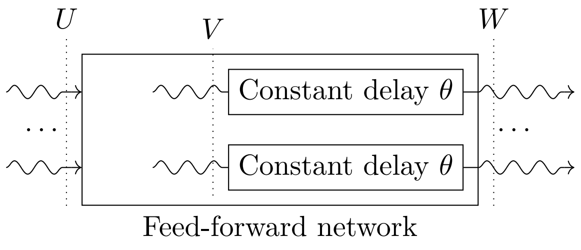


Figure B.1: Illustration of the proof principle. The network is feed-forward between U and W, and V is exactly θ seconds before W.

We fix some θ such that $0 < \theta < T^{\min}$ with $T^{\min}$ the minimum propagation time of the links. We consider V, the point that is exactly θ seconds before W. This point is on the same transmission link as W because t_{prop} is the minimal propagation delay of the links. We consider the true network, i.e. with U and W connected together. In the rest of the proof, take any $\tau \geq 0$.

For $M = U, V, W$ call $\boldsymbol{R}_M$ [resp. $\boldsymbol{R}^{\tau}_M(t)$] the vector of cumulative arrival functions [resp. stopped at time τ i.e., $\boldsymbol{R}^{\tau}_M(t) = \min(\boldsymbol{R}_M(t), \boldsymbol{R}_M(\tau))$]. Also for $M = V, W$ call $\boldsymbol{R}'^{\tau}_M$ the vector of cumulative arrival functions that are obtained at points V, W when the inputs at U are stopped at time τ, i.e. $\boldsymbol{R}'^{\tau}_M(t) = \min(\boldsymbol{R}_M(t), \boldsymbol{R}_U(\tau))$. Last, call $\boldsymbol{b}^{\tau}_M$ and $\boldsymbol{b}'^{\tau}_M$ the corresponding best burst bounds. For example, for any component i of vector $\boldsymbol{b}'^{\tau}_M$, $b'^{\tau}_{M,i} = \sup_{t' \geq t}(R'^{\tau}_{M,i}(t') - R'^{\tau}_{M,i}(t) - r(t'-t))$, where r is the leaky-bucket rate of the flow corresponding to component i.

Lemma B.1
At point V, $\boldsymbol{b}^{\tau}_V \leq \boldsymbol{b}'^{\tau}_V$

Proof of Lemma B.1. Note that for $t \leq \tau, \boldsymbol{R}^{\tau}_M(t) = \boldsymbol{R}'^{\tau}_M(t)$ and for $t > \tau, \boldsymbol{R}^{\tau}_M(t)$ is a constant. The result is thus obtained by splitting the sup of the definition of $\boldsymbol{b}'^{\tau}_V$, $\boldsymbol{b}^{\tau}_V$ into the three options: either $t, t' \leq \tau$, or $t \leq \tau, t' > \tau$ or $t, t' > \tau$. □

As $\mathcal{FF}$ computes a bound on the output bursts for a given input, we know that $\boldsymbol{b}'^{\tau}_W \leq \mathcal{FF}(\boldsymbol{b}^{\tau}_U)$. As the delay between V and W is constant equal to θ, a change of variable $(s, s') \leftarrow (t + \theta, t' + \theta)$ in the definition of $\boldsymbol{b}'^{\tau}_V$ gives $\boldsymbol{b}'^{\tau}_V = \boldsymbol{b}'^{\tau}_W \leq \mathcal{FF}(\boldsymbol{b}^{\tau}_U)$. Using Lemma 1, we obtain:

$$\forall \tau \geq 0, \boldsymbol{b}^{\tau}_V \leq \mathcal{FF}(\boldsymbol{b}^{\tau}_U) \tag{B.1}$$

Using $\boldsymbol{R}_W(t + \tau) = \boldsymbol{R}_V(t)$ for any $t \geq 0$, we obtain $\boldsymbol{b}^{\tau+\theta}_W = \boldsymbol{b}^{\tau}_V$. Also, as W and U are connected together, Equation B.1 gives $\boldsymbol{b}^{\tau+\theta}_U = \boldsymbol{b}^{\tau+\theta}_W = \boldsymbol{b}^{\tau}_V \leq \mathcal{FF}(\boldsymbol{b}^{\tau}_U)$. Apply this to $\tau = k\theta$ for $k \in \mathbb{N}$ and obtain:

$$\forall k \in \mathbb{N}, \boldsymbol{b}^{(k+1)\theta}_U \leq \mathcal{FF}(\boldsymbol{b}^{k\theta}_U) \tag{B.2}$$

The network is empty at $t = 0$ so $\boldsymbol{b}^0_U = \boldsymbol{0} \leq \bar{\boldsymbol{b}}$. Now $\mathcal{FF}$ can be assumed to be wide-sense increasing as per [Bouillard, Boyer, Le Corronc 2018, Chap. 12]. By monotonicity of $\mathcal{FF}$, the fact that $\mathcal{FF}(\bar{\boldsymbol{b}}) = \bar{\boldsymbol{b}}$ and a simple induction argument, it follows that $\boldsymbol{b}^{k\theta}_U \leq \bar{\boldsymbol{b}}$ for all $k \in \mathbb{N}$. For any $\tau \in [0, +\infty)$, we have $\boldsymbol{b}^{\tau}_U \leq \boldsymbol{b}^{k\theta}_U$ with $k = \lceil \frac{\tau}{\theta} \rceil$, thus $\boldsymbol{b}^{\tau}_U \leq \bar{\boldsymbol{b}}$ for all $\tau \geq 0$. Now $\boldsymbol{b}_U = \sup_{\tau \geq 0} \boldsymbol{b}^{\tau}_U$, thus $\bar{\boldsymbol{b}}$ is a finite bound for $\boldsymbol{b}_U$ and the network is stable. □

B.2 Proofs of Chapter 4

B.2.1 Network Calculus Results for Non-Lossless Non-FIFO systems

In this appendix, we provide several classic network-calculus results that remain valid for non-lossless, non-FIFO networks.

Lemma B.2 (Arrival curve of a flow at the output of a system with bounded delay)
Consider a flow f entering a system $\mathcal{S}$. Assume that each bit of f that exits $\mathcal{S}$ suffers a delay

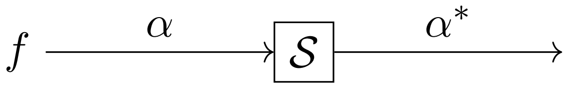


Figure B.2: Notations for Appendix B.2.1. Flow f with input arrival curve α enters system $\mathcal{S}$ and exits with an output arrival curve α^*.

within $\mathcal{S}$ that is bounded within $[d, D]$. Finally assume that α is an arrival curve for f at the input of $\mathcal{S}$. $\mathcal{S}$ does not need to be FIFO or lossless.

Then, $\alpha^* = \alpha \oslash \delta_{D-d}$ is an arrival curve for f at the output of $\mathcal{S}$.

Proof. Denote by R_f the cumulative process of f at the input of $\mathcal{S}$ (Figure B.2). We decompose $R_f = R_a + R_b$, with R_a the cumulative process, at the input, for the stream of bits of f that are not lost inside $\mathcal{S}$ and $R_b = R_f - R_a$. $R_a(t)$ is defined as the number of bits of f that eventually exit $\mathcal{S}$ (that are not lost inside it) and that are observed at the input of $\mathcal{S}$ during the interval $[0, t]$. Note that the cumulative functions R_a and R_b are unknown in general: When a bit is observed at the input of $\mathcal{S}$, the real-life observer cannot infer whether it will be lost within $\mathcal{S}$ or not. However, we can still work on the unknown functions R_a and R_b.

We denote by R^*_a [resp., R^*_b] the output cumulative function related to the input process R_a [resp., R_b]. Hence, $R^*_a(t)$ is defined as the number of bits of f that exit $\mathcal{S}$ (are not lost inside it) and that are seen at the output of $\mathcal{S}$ during the interval $[0, t]$.

All cumulative functions are positive, wide-sense increasing and defined for $t \geq 0$ (Section 2.2). We extend their definition domain by using the convention that all cumulative functions equal zero in $\mathbb{R}-$: $\forall t \leq 0, R_a(t) = R_b(t) = R^*_a(t) = R^*_b(t) = 0$.

As α is an arrival curve for f at the input of $\mathcal{S}$, for all $s \leq t$,

$$\begin{aligned} R_f(t) - R_f(s) &\leq \alpha(t-s) \\ R_a(t) + R_b(t) - R_a(s) + R_b(s) &\leq \alpha(t-s) \\ R_a(t) - R_a(s) &\leq \alpha(t-s) + R_b(s) - R_b(t) \end{aligned}$$

As $s \leq t$, and R_b is wide-sense increasing, $R_b(s) - R_b(t) \leq 0$ and $R_a(t) - R_a(s) \leq \alpha(t-s)$ which shows that the cumulative process R_a is also α-constrained.

By definition of R_b and R^*_b, $\forall t, R^*_b(t) = 0$ and

$$R^*_a(t) = R^*_f(t) \tag{B.3}$$

The system $\mathcal{S}$ is not FIFO but the non-lost bits have a maximum delay of D. Hence, all the bits of R_a that have entered $\mathcal{S}$ at t have exited $\mathcal{S}$ by $t + D$,

$$R^*_a(t + D) \geq R_a(t) \tag{B.4}$$

Equation (B.4) is valid for $t < 0$ because $R_a(t) = 0$ for $t < 0$ and R^*_a is a positive function. Similarly, the minimum delay of each data unit within $\mathcal{S}$ is d. As such, all the data units that have exited $\mathcal{S}$ by $t + d$ must have entered $\mathcal{S}$ before t,

$$R^*_a(t + d) \leq R_a(t) \tag{B.5}$$

Equation (B.5) is again valid for $t < 0$: $R_a(t) = 0$ but $R^*_a(t + d)$ also equals zero because the minimum time that a bit needs to reach the output is d.

Then, $\forall t \geq s$,

$$\begin{aligned} & R^*_a(t) - R^*_a(s) \\ & \quad \leq R_a(t-d) - R_a(s-D) \qquad \triangleright \text{(B.4) and (B.5)} \\ & \quad \leq \alpha(t - s + (D-d)) \qquad \triangleright R_a \text{ is } \alpha\text{-constrained} \\ & \quad \leq (\alpha \oslash \delta_{D-d})(t-s) \end{aligned}$$

Combining the above result with (B.3) shows, $\forall t \geq s$

$$R^*_f(t) - R^*_f(s) \leq (\alpha \oslash \delta_{D-d})(t-s) \tag{B.6}$$

which proves that $\alpha^* = \alpha \oslash \delta_{D-d}$ is an arrival curve for f at the output of $\mathcal{S}$. □

For a system with a constant delay or without any delay, Lemma B.2 is simplified as follows.

Lemma B.3 (A system with constant delay keeps the arrival curves)
If $\mathcal{S}$ is a system in which the non-lost bits of any flow have a constant delay, then an arrival curve for a flow or aggregate of flows at the input of $\mathcal{S}$ is also an arrival curve for the same flow or aggregate of flows at the output of $\mathcal{S}$.

Proof. For the aggregate, we simply need to consider the whole aggregate as a unique flow when applying Lemma B.2. □

B.2.2 Proof of Theorem 4.1

Proof of Theorem 4.1. Consider a vertex n and a flow f such that n contains a PEF for f, noted $\texttt{PEF}_n(f)$. From the device model of Chapter 2, Section 2.6, we first note that flow f is packetized at both the input and the output of $\texttt{PEF}_n(f)$.

Proof of Item 1/ As per the model in Section 4.2.2, $\texttt{PEF}_n(f)$ is a network element that can lose packets but does not have any delay for the forwarded packets. As a consequence, it does not have any delay for the forwarded bits either (both its input and its output are packetized). Applying Lemma B.3 proves that an arrival curve for f at the input of the PEF, $\alpha_{f,\text{PEF}^{\text{in}}}$, is also an arrival curve for f at the output of the PEF.

Item 2/ Consider a diamond ancestor a of n. The observation point a^* is located at the output of the input port within a. As such, flow f is packetized at the observation point a^*. As such, a bound on the per-bit delay between a^* and either the PEF's input or the PEF's output $\texttt{PEF}^*$ is also a per-packet delay bound on the delay between the same two observation points, and *vice versa*.

Denote by $\mathcal{P}^{\mathcal{G}(f)}_{a,n}$ the set of all possible paths from a to n in $\mathcal{G}(f)$ and consider a data unit m of f such that m is not lost for n. By definition of the diamond ancestor, a is not an EP-vertex for f, thus the data unit m is observed exactly once at a.

Denote by $\{P^m_i\}_{i\in\mathcal{I}(m)}$ the set of packets containing m that reach $\texttt{PEF}_n(f)$, with $\mathcal{I}(m)$ a set to index them. $\mathcal{I}(m)$ is not empty because m is not lost for n (at least one packet

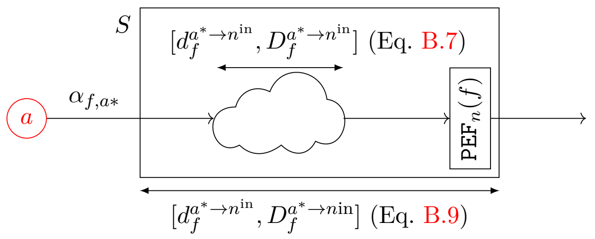

S
$[d_f^{a^*\to n^{\text{in}}}, D_f^{a^*\to n^{\text{in}}}]$ (Eq. B.7)
a
$\alpha_{f,a*}$
$\text{PEF}_n(f)$
$[d_f^{a^*\to n^{\text{in}}}, D_f^{a^*\to n\text{in}}]$ (Eq. B.9)

Figure B.3: Notations for Appendix B.2.2: System from diamond ancestor a to the PEF, focusing on the non-lost data units.

containing m reaches n). Furthermore, $\mathcal{I}(m)$ is a finite set because $\mathcal{G}(f)$ is finite and acyclic: m is replicated a finite number of times.

For i in $\mathcal{I}(m)$, call ρ_i the path within $\mathcal{G}(f)$ that the packet P_i^m took from the source of f to n. By definition of a diamond ancestor of n, this path crosses a and by definition of $\mathcal{P}_{a,n}^{\mathcal{G}(f)}$, there exists a path $p_i \in \mathcal{P}_{a,n}^{\mathcal{G}(f)}$ such that packet P_i^m took path p_i between a and n (p_i is a sub-path of ρ_i).

Denote by d_i^m the delay of packet P_i^m between the output of a and the input of $\texttt{PEF}_n(f)$. By definition of the notations $D_f^{a^*\to n^{\text{in}}}$ and $d_f^{a^*\to n^{\text{in}}}$ used in Theorem 4.1,

$$d_f^{a^*\to n^{\text{in}}} \leq d_i^m \leq D_f^{a^*\to n^{\text{in}}} \tag{B.7}$$

The values $d_f^{a^*\to n^{\text{in}}}$ and $D_f^{a^*\to n^{\text{in}}}$ can be seen as the lower and upper-bound of the non-lost data units through the system located between the output of a and the input of $\texttt{PEF}_n(f)$. This system is represented with a cloud in Figure B.3.

The data unit m exits $\texttt{PEF}_n(f)$ as soon as one of the the packets $\{P_i^m\}_{i\in\mathcal{I}(m)}$ reaches the packet-elimination function. If we denote by $d_m^{a^*\to\texttt{PEF}^*}$ the delay of the data unit m from the output of a to the output of the PEF, we have, $\forall m \in f$, m not lost for n,

$$\exists i \in \mathcal{I}(m), \quad d_m^{a^*\to\texttt{PEF}^*} = d_i^m \tag{B.8}$$

Combining Equations (B.7) and (B.8) gives

$$\forall m \in f, m \text{ not lost for } n, \quad d_f^{a^*\to n^{\text{in}}} \leq d_m^{a^*\to\texttt{PEF}^*} \leq D_f^{a^*\to n^{\text{in}}} \tag{B.9}$$

Equation (B.9) proves that any non-lost data units of f for n suffer through the system S in Figure B.3 a delay bounded in $[d_f^{a^*\to n^{\text{in}}}, D_f^{a^*\to n^{\text{in}}}]$. As both a^* and the output of the PEF are packetized, this also proves that each bit of f that is not lost within $\mathcal{S}$ (neither in the cloud of Figure B.3 nor in the PEF) suffers a delay through $\mathcal{S}$ that is bounded within $[d_f^{a^*\to n^{\text{in}}}, D_f^{a^*\to n^{\text{in}}}]$. We apply Lemma B.2 and obtain that $\alpha_{f,a^*} \oslash \delta_{(D_f^{a^*\to n^{\text{in}}})-(d_f^{a^*\to n^{\text{in}}})}$ is an arrival curve for f at the output of the PEF.

The last sentence of the theorem is a direct application of Theorem 2.1. □

B.2.3 Proof of Corollary 4.1

Proof of Corollary 4.1. The replication is performed by the switching fabric, both its input and output are hence packetized. The same remark applies for the PEF thus we conclude that both the input and the output of each system S_i is also packetized. Therefore, the per-bit delay of flow f through S_i is also bounded by $[d_i, D_i]$.

For each $i \in [\![1, N]\!]$, the application of Lemma B.2 gives the arrival curve for f at the output of S_i

$$\forall \in [\![1, N]\!], \quad \alpha_{f,S_i^*} = \alpha_f \oslash \delta_{D_i - d_1}$$

We then obtain $\alpha_{f,\mathrm{PEF}^{\mathrm{in}}}$

$$\begin{aligned}\alpha_{f,\mathrm{PEF}^{\mathrm{in}}} &= \sum_{i\in[\![1,n]\!]} \alpha_{f,S_i^*}\\ &= \sum_{i\in[\![1,n]\!]} \alpha_f \oslash \delta_{D_i - d_1}\end{aligned}$$

We then apply Equation (4.1) with a^* being the input of the replication function in Figure 4.13. A lower delay bound for f from the ancestor a to the input of the PEF along any possible paths (*i.e.*, through any S_i) is

$$d_f^{a^* \to n^{\mathrm{in}}} = \min\{d_i; i \in [\![1, N]\!]\}$$

Similarly,

$$D_f^{a^* \to n^{\mathrm{in}}} = \max\{D_i; i \in [\![1, N]\!]\}$$

and (4.1) can be written

$$\begin{aligned}\alpha_f^{a^* \to n^{\mathrm{in}}} &= \alpha_{f,a^*} \oslash \delta_{D_f^{a^* \to n^{\mathrm{in}}} - d_f^{a^* \to n^{\mathrm{in}}}}\\ &= \alpha_f \oslash \delta_{\max_i D_i - \min_i d_i}\end{aligned}$$

We apply Theorem 4.1: $\alpha_{f,\mathrm{PEF}^*} = \alpha_{f,\mathrm{PEF}^{\mathrm{in}}} \otimes \alpha_f^{a^* \to n^{\mathrm{in}}}$ is an arrival curve for f at the output of the PEF. Replacing with the above expressions for $\alpha_{f,\mathrm{PEF}^{\mathrm{in}}}$ and $\alpha_f^{a^* \to n^{\mathrm{in}}}$ gives Equation (4.3) of Corollary 4.1. □

B.2.4 Proof of Proposition 4.1

Proof of Proposition 4.1. Take a leaky-bucket arrival curve $\gamma_{r,b}$. And $[d_1, D_1]$, $[d_2, D_2]$ two intervals of $\mathbb{R}^+$.

We first prove the result when $d_2 - D_1 \geq b/r$, we prove the other situation afterwards.

Case $d_2 - D_1 \geq b/r$:

Applying Corollary 4.1 with $N = 2$ systems S_1, S_2 with bounded intervals $[d_1, D_1]$ and $[d_2, D_2]$ gives that

$$\begin{aligned}\alpha^* &= \left(\sum_{i\in[\![1,2]\!]} \gamma_{r,b} \oslash \delta_{D_i - d_i}\right) \otimes (\gamma_{r,b} \oslash \delta_{D_2 - d_1})\\ &= \left(\gamma_{r,b+r(D_1-d_1)} + \gamma_{r,b+r(D_2-d_1)}\right) \otimes \gamma_{r,b+r(D_2-d_1)}\\ &= \gamma_{2r,2b+r(D_1-d_1+D_2-d_2)} \otimes \gamma_{r,b+r(D_2-d_1)}\end{aligned} \tag{B.10}$$

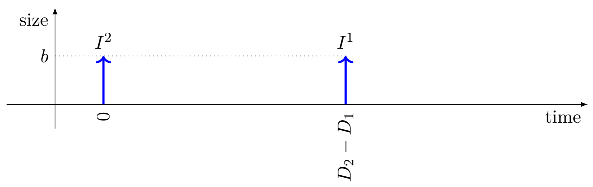


Figure B.4: Source output in the trajectory that achieves the tightness of Corollary 4.1. Two "initiator" data units are sent at 0 and at $D_2 - D_1$.

Table B.1: Generation of the Data-Units of Category I in the Trajectory that Achieves the Tightness of Corollary 4.1.

Data unit m	Size, size(m)	Generation time, $\mathcal{G}(m)$
I^2	b	0
I^1	b	$D_2 - D_1$

is an arrival curve for f after the PEF in Figure 4.13.

In the following, we exhibit a trajectory with a $\gamma_{r,b}$-constrained source for f and no minimal packet length. We exhibit also two systems S_1, S_2, in which the delay of the non-lost data-units is in the intervals $[d_1, D_1]$ and $[d_2, D_2]$. The proof operates in several steps, as follows:

Definition of several constants used in the proof

We define

$$\chi^1 \triangleq \left\lceil \frac{r(D_1 - d_1)}{b} \right\rceil \quad \text{and} \quad \chi^2 \triangleq \left\lceil \frac{r(D_2 - d_2)}{b} \right\rceil \tag{B.11}$$

(Note that $\chi^1 \geq 1$ and $\chi^2 \geq 1$)

Last, we define

$$\psi = \left\lceil \frac{r(d_2 - D_1) - b}{b} \right\rceil \tag{B.12}$$

and we also have $\psi \geq 1$.

Description of the traffic generation at the source

For the sake of clarity, we classify the data-units generated by the source into four categories: I, B, S and X. The category of a data-unit defines the role that the data-unit has in the trajectory. Each of the three first categories (I, B, S) has two sub-categories that we distinguish by using a superscript (e.g., I^1 and I^2). This sub-category notion is used in order to distinguish the role of every system (S_1 or S_2) in the trajectory.

Subcategories do not infer the order with which data-units are generated. The notions of categories and subcategories are only used in the proof, they are not related to any physical property of the packets.

Category I: The source generates two "initiator" data-units: I^2 [resp., I^1] at absolute time 0 [resp., $(D_2 - D_1)$], of length b (see Table B.1). Figure B.4 shows the timeline of the data-units I out of the source.

Table B.2: Generation of the Data-Units of Category B in the Trajectory that Achieves the Tightness of Corollary 4.1.

Data unit m		Size, size(m)	Generation time, $\mathcal{G}(m)$
$\forall k \in [\![1, \chi^2 - 1]\!]$,	B^2_k	b	$k\frac{b}{r}$
	$B^2_{\chi^2}$	$r(D_2 - d_2) - (\chi^2 - 1)b$	$(D_2 - d_2)$
$\forall k \in [\![1, \chi^1 - 1]\!]$,	B^1_k	b	$(D_2 - D_1) + k\frac{b}{r}$
	$B^1_{\chi^1}$	$r(D_1 - d_1) - (\chi^1 - 1)b$	$(D_2 - D_1) + (D_1 - d_1)$

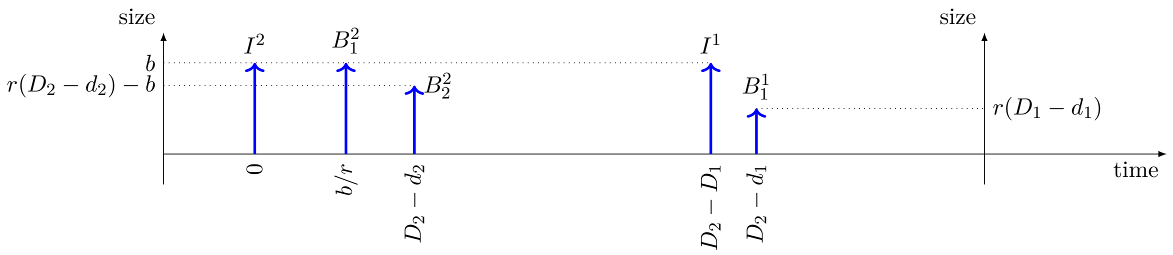


Figure B.5: Example of the source output in the trajectory that achieves the tightness, focusing on categories I and B.

Note: The role of the two data-units of category I is to initiate the backlog period. In the next parts of the proof, we create a situation where I^1 and I^2 exit the PEF of Figure 4.13 at the same time, creating the $2b$ part of the burst in the term $\gamma_{2r,\mathbf{2b}+r(D_1-d_1+D_2-d_2)}$ of (B.10).

Category B: In addition to the data-units of category I, the source generates χ^2 data-units of subcategory B^2 and χ^1 data-units of subcategory B^1, as described in Table B.2. A possible output of the source when combining categories I and B is shown in Figure B.5. In the proposed situation, we have $\chi_1 = 1$ (*i.e.*, $r(D_1 - d_1) \leq b$). Then the interval $[\![1, \chi^1 - 1]\!]$ in Table B.2 is empty and category B^1 contains a unique data-unit $B^1_1 = B^1_{\chi^1}$ of size $r(D_1 - d_1)$ and sent at time $(D_2 - D_1) + (D_1 - d_1) = D_2 - d_1$. In Figure B.5, χ^2 equals 2, and category B_2 is made of two data-units: B^2_1, of size b, released at time b/r; and B^2_2, of size $r(D_2 - d_2) - b$, released at time $(D_2 - d_2)$.

Note that for any value of χ^1, χ^2, by Table B.2,

$$\forall j \in \{1, 2\} \qquad \sum_{k \in [\![1, \chi^j]\!]} \text{size}(B^j_k) = r(D_j - d_j) \tag{B.13}$$

Note: The role of the data-units of category B is to participate in the burst term of $\gamma_{2r,2b+r(D_1-d_1+D_2-d_2)}$ in (B.10). In the next parts of the proof, we create a situation where all data-units of category B (both subcategories B^1 and B^2) are released at the same time, simultaneously with data-units I^1 and I^2. This give the part $r(D_1 - d_1 + D_2 - d_2)$ in the burst term of $\gamma_{2r,2b+\mathbf{r(D_1-d_1+D_2-d_2)}}$.

We now prove that data-units of subcategory B^1 [resp., B^2] are generated after data-units of subcategory I^1 [resp., I^2] and in the order of their lower-script index.

• If $\chi^2 = 1$, then the first data-unit of B^2 is sent at $D_2 - d_2$ and $D_2 - d_2 \geq 0$, so data-unit of B^2 is generated after the data-unit of I^2.

• If $\chi^2 \geq 1$, then the first data-unit of B^2 is sent at $b/r \geq 0$, so data-units of B^2 are

Table B.3: Generation of the Data-Units of Category S in the Trajectory that Achieves the Tightness of Corollary 4.1.

Data unit m		Size, size(m)	Generation time, $\mathcal{G}(m)$
$\forall k \in [\![1, \psi - 1]\!]$,	S^2_k	b	$(D_2 - d_2) + k\frac{b}{r}$
	S^2_ψ	$r(d_2 - D_1) - \psi b$	$(D_2 - D_1 - \frac{b}{r})$
$\forall k \in [\![1, \psi - 1]\!]$,	S^1_k	b	$(D_2 - d_1) + k\frac{b}{r}$
	S^1_ψ	$r(d_2 - D_1) - \psi b$	$(D_2 + d_2) - (D_1 + d_1) - \frac{b}{r}$

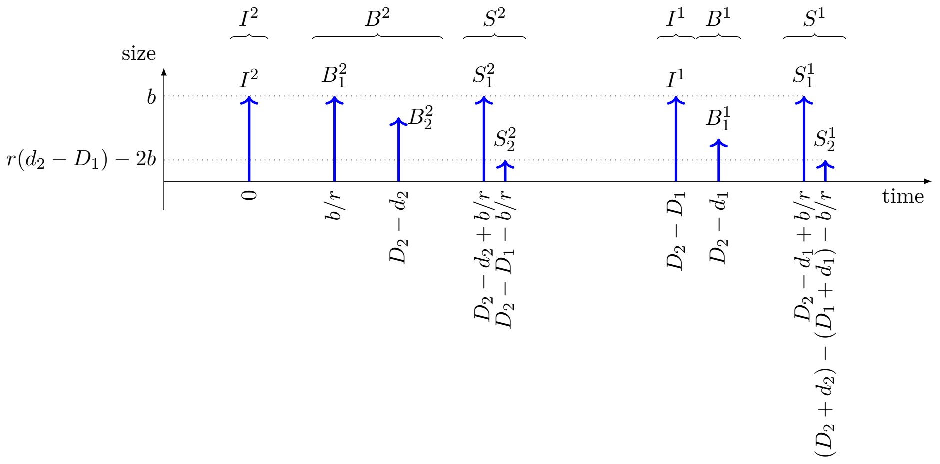


Figure B.6: Source output, with the three categories I, B and S of data-units

generated after the data-units of I^2. Also, the data-units $(B^2_k)_{\kappa \in [\![1,\chi^2]\!]}$ are generated in the same order as their index: this is clear for indexes up to $\chi^2 - 1$. For the order between $B^2_{\chi^2-1}$ and $B^2_{\chi^2}$, we note that $\left\lceil \frac{r(D_2-d_2)}{b} \right\rceil \frac{b}{r} - \frac{b}{r} \le D_2 - d_2$ by property of the ceiling function.

• If $\chi^1 = 1$, then the first data-unit of B^1 is sent at $(D_2 - d_1)$ thus after the data-unit of I^1 (as $D_1 \ge d_1$).

• If $\chi^1 \ge 1$, then by using the same reasoning for B^2, we obtain that data-units of B^1 are generated after the initiator I^1 and they are released in the order of their lower-script index.

Category S: In addition to the data-units of categories I and B, the source generates ψ data-units of subcategory S^2 and ψ data-units of subcategory S^1, as described in Table B.3. A possible output of the source when adding category S to Figure B.5 is shown in Figure B.6. In the proposed situation, we have $\psi = 2$ and subcategories S^1 and S^2 are both made of two data-units: S^1_1 [resp., S^2_1], of size b, released $\frac{b}{r}$ after the last data-unit of B^1 [resp., B^2] and S^1_2 [resp., S^2_2], of size $r(d_2 - D_1) - 2b$, released at $(D_2 + d_2) - (D_1 + d_1) - \frac{b}{r}$ [resp., $D_2 - D_1 - \frac{b}{r}$].

Note that for any value of ψ, by Table B.3,

$$\forall j \in \{1, 2\} \qquad \sum_{k \in [\![1,\psi]\!]} \mathrm{size}(S^j_k) = r(d_2 - D_1) - b \tag{B.14}$$

Note: The role of the data-units of category S is to participate in the peak-rate term of $\gamma_{\mathbf{2r},2b+r(D_1-d_1+D_2-d_2)}$ in (B.10). The output traffic should maintain the peak rate for a sufficient duration so that the obtained cumulative output intersects with the curve $\gamma_{r,b+r(D_2-d_1)}$.

Table B.4: Generation of the Data-units of Category X in the Trajectory that Achieves the Tightness of Corollary 4.1.

Data unit m	Size, size(m)	Generation time, $\mathcal{G}(m)$
$\forall k \in \mathbb{N}^*$, X_k	b	$(D_2 + d_2) - (D_1 + d_1) + (k-1)\frac{b}{r}$

In the next parts of the proof, we create a situation where each data unit of subcategory S^1 *is released at the same time as its peer of subcategory* S^2*. This creates a peak rate* $2r$ *for a duration of at least* $d_2 - D_1 - b/r$*. The resulting cumulative output intersects the curve* $\gamma_{r,b+r(D_2-d_1)}$*.*

We now check the order of the data data units of category S.

• If $\psi = 1$, then the first data-unit of S^2 is sent at $D_2 - D_1 - b/r$, whereas the last data-unit of B^2 was sent at $D_2 - d_2$. By assumption, $d_2 - D_1 \geq b/r$, so $D_2 - D_1 - b/r \geq D_2 - d_2$ and the first data-unit of S^2 is sent after the last data-unit of B^2.

• If $\psi \geq 1$, then the first data-unit of S^2 is sent b/r after the last data-unit of B^2. Also, the data-units $(S^2_k)_{\kappa \in [\![1,\psi]\!]}$ are generated in the same order as their index: this is clear for indexes up to $\psi - 1$. For the order between $S^2_{\psi-1}$ and S^2_ψ, we note that $(\psi-1)\frac{b}{r} \leq (d_2 - D_1) - \frac{b}{r}$ by properties of the ceiling function and so $(D_2 - d_2) + (\psi - 1)\frac{b}{r} \leq (D_2 - d_2) + (d_2 - D_1) - \frac{b}{r}$, *i.e.*, $D_2 - d_2 + (\psi-1)\frac{b}{r} \leq D_2 - D_1 - \frac{b}{r}$, *i.e*, $S^2_{\psi-1}$ is sent before S^2_ψ

We then apply the same principles for S^1. We thus prove that data-units of subcategory S^1 [resp., S^2] are generated after data-units of subcategory B^1 [resp., B^2] and in the order of their index. We also observe that the last data-unit of subcategory S^2 is sent at $D_2 - D_1 - \frac{b}{r}$, whereas the data-unit of subcategory I^1 is sent at $D_2 - D_1$, hence data-units of subcategory S^2 are sent before the data-unit of subcategory I^1.

Category X: After the data-units of subcategory S^1, the source generates for eternity data-units $(X_n)_{n\in\mathbb{N}^*}$ of size b with a period b/r (see Table B.4). The first one of these data-units is sent b/r after the last data-unit of S^1. Figure B.7 presents the output of the source with all four categories.

Note: The role of category X *is to generate the sustained rate term in the curve* $\gamma_{\mathbf{r},b+r(D_2-d_1)}$ *in (B.10).*

Properties of the traffic generation at the source

Now that we have described the profile of the traffic generated by the source, we can prove that the generation is $\gamma_{r,b}$-compliant. This is done with the following lemmas.

Lemma B.4 (Size of the data-units)
For any data-unit P *described in the previous Subsection,* $0 \leq size(P) \leq b$

Proof of Lemma B.4. We focus on certain data-units.

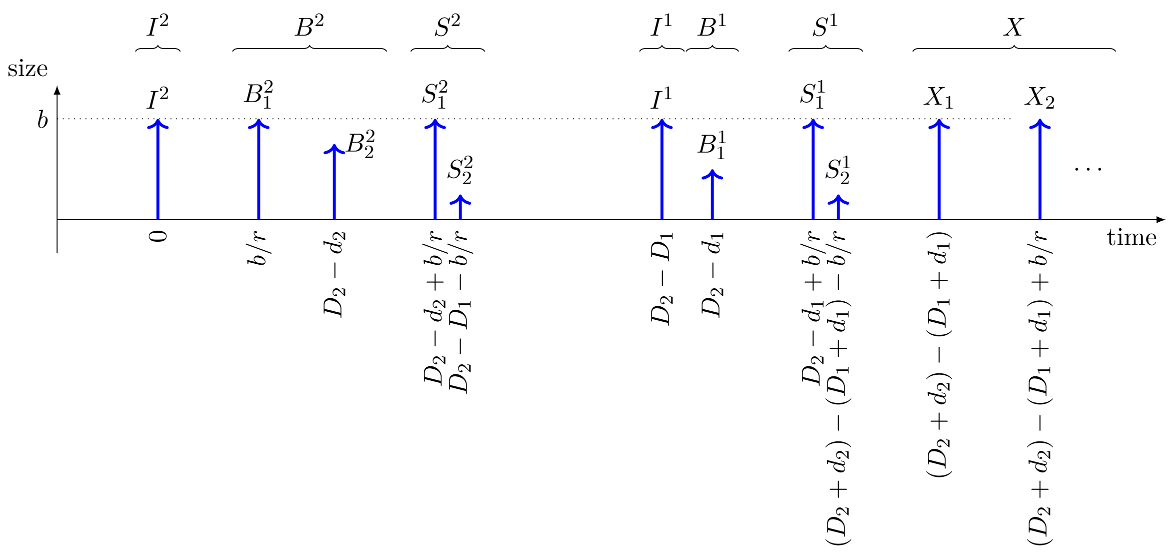


Figure B.7: Source output, with the four categories of data units: I, B, S and X of data-units, in the trajectory that achieves the tightness of Corollary 4.1.

- $B^2_{\chi^2}$: By property of the ceiling function:

$$\begin{aligned}
\frac{r(D_2-d_2)}{b} \le \chi^2 & \le \frac{r(D_2-d_2)}{b} + 1 \\
r(D_2-d_2) - b \le (\chi^2 - 1)b & \le r(D_2-d_2) \qquad \triangleright\, b > 0 \\
-r(D_2-d_2) + b \ge -(\chi^2 - 1)b & \ge -r(D_2-d_2) \\
b \ge r(D_2-d_2) - (\chi^2 - 1)b & \ge 0
\end{aligned}$$

- $B^1_{\chi^1}$: same idea
- S^1_ψ and S^2_ψ (they have the same size): By property of the ceiling function:

$$\begin{aligned}
\frac{r}{b}(d_2 - D_1) - 1 \le \psi & \le \frac{r}{b}(d_2 - D_1) \\
-r(d_2 - D_1) + b \ge -\psi b & \ge -r(d_2 - D_1) \qquad \triangleright\, b > 0 \\
b \ge r(d_2 - D_1) - \psi b & \ge 0
\end{aligned}$$

All the other data-units have the same size, equal to the burst b. □

Lemma B.5 (Minimum time distance between two successive data-units)
Consider two successive data-units m, m' in the traffic described in the previous Subsection, i.e., *m' is the first data-unit sent after m. Note $\mathcal{G}(m)$ and $\mathcal{G}(m')$ the time at which they are generated.*

Then $\mathcal{G}(m') - \mathcal{G}(m) \ge \frac{size(m')}{r}$

Proof of Lemma B.5. We simply describe all the possible combinations:

- Case $m = I^2$ and $m' = B^2_1$:

We have $\mathcal{G}(m) = 0$ (see Table B.1). If $\chi^2 = 1$, then $\mathcal{G}(m') - \mathcal{G}(m) = (D_2 - d_2)$, size$(m') =$

$r(D_2 - d_2)$ (see Table B.2) and the result holds. If $\chi^2 \geq 2$, then $\mathcal{G}(m') - \mathcal{G}(m) = b/r$, $\text{size}(m') = b$ and the result holds.

• Case $m, m' \in B^2$ (when $\chi^2 \geq 2$):

There exists $k \in [\![1, \chi^2 - 1\ ]\!]$ such that $m = B^2_k$ and $m' = B^2_{k+1}$. If $k \leq \chi^2 - 2$, then $\mathcal{G}(m') - \mathcal{G}(m) = b/r$, $\text{size}(m') = b$ and the result holds. If $k = \chi^2 - 1$, then:

$$\begin{aligned}\mathcal{G}(m') - \mathcal{G}(m) &= (D_2 - d_2) - k\frac{b}{r}\\ &= (D_2 - d_2) - (\chi^2 - 1)\frac{b}{r}\\ &= r \cdot \text{size}(m')\end{aligned}$$

• Case $m = B^2_{\chi^2}$ and $m' = S^2_1$:

We have $\mathcal{G}(m) = D_2 - d_2$ (Table B.2). If $\psi = 1$, then $m' = S^2_\psi$ and

$$\begin{aligned}\mathcal{G}(m') - \mathcal{G}(m) &= D_2 - D_1 - \frac{b}{r} - (D_2 - d_2)\\ &= d_2 - D_1 - \frac{b}{r}\\ &= \frac{r(d_2 - D_1) - \psi b}{r}\\ &= \text{size}(m')/r\end{aligned}$$

If $\psi \geq 2$, then $\mathcal{G}(m') - \mathcal{G}(m) = b/r$ (see Tables B.2 and B.3) and $\text{size}(m') = b$ so the result holds.

• Case $m, m' \in S^2$ (when $\psi \geq 2$):

There exists $k \in [\![1, \psi - 1]\!]$ such that $m = S^2_k$ and $m' = S^2_{k+1}$. If $k \leq \psi - 2$, then $\mathcal{G}(m') - \mathcal{G}(m) = b/r$ and $\text{size}(m') = b$ so the result holds. If $k = \psi - 1$, then

$$\begin{aligned}\mathcal{G}(m') - \mathcal{G}(m) &= d_2 - D_1 - \psi\frac{b}{r}\\ &= \text{size}(S^2_\psi)/r = \text{size}(m')/r\end{aligned}$$

• Case $m = S^2_\psi$ and $m' = I^1$:

Then we have $\mathcal{G}(m) = D_2 - D_1 - \frac{b}{r}$ (Table B.3) and $\mathcal{G}(m') = (D_2 - D_1)$ (Table B.1). So $\mathcal{G}(m') - \mathcal{G}(m) = b/r = \text{size}(m')/r$.

• Case $m = I^1$ and $m' \in B^1$:

We have $\mathcal{G}(m) = (D_2 - D_1)$ (see Table B.1). If $\chi^1 = 1$, then $\mathcal{G}(m') - \mathcal{G}(m) = (D_1 - d_1)$, $\text{size}(m') = r(D_1 - d_1)$ (see Table B.2) and the result holds. If $\chi^1 \geq 2$, then $\mathcal{G}(m') - \mathcal{G}(m) = b/r$, $\text{size}(m') = b$ and the result holds also.

• Case $m, m' \in B^1$ (when $\chi^1 \geq 2$):

There exists $k \in [\![1, \chi^1 - 1\ ]\!]$ such that $m = B^1_k$ and $m' = B^1_{k+1}$. If $k \leq \chi^1 - 2$, then

$\mathcal{G}(m') - \mathcal{G}(m) = b/r$, $\text{size}(m') = b$ and the result holds. If $k = \chi^1 - 1$, then

$$\begin{aligned}\mathcal{G}(m') - \mathcal{G}(m) &= (D_2 - D_1) + (D_1 - d_1) - (D_2 - D_1) - k\frac{b}{r} \\ &= (D_1 - d_1) - (\chi^1 - 1)\frac{b}{r} \\ &= r \cdot \text{size}(m')\end{aligned}$$

- Case $m = B^1_{\chi^1}$ and $m' = S^1_1$:

  We have $\mathcal{G}(m) = (D_2 - d_1)$ (Table B.2). If $\psi = 1$, then $m' = S^1_\psi$ and

$$\begin{aligned}\mathcal{G}(m') - \mathcal{G}(m) &= D_2 + d_2 - D_1 - d_1 - b/r - D_2 + d_1 \\ &= d_2 - D_1 - b/r \\ &= \text{size}(m')/r\end{aligned}$$

If $\psi \geq 2$, then $\mathcal{G}(m') - \mathcal{G}(m) = b/r$ (see Tables B.2 and B.3) and $\text{size}(m') = b$ so the result holds.

- Case $m, m' \in S^1$ (when $\psi \geq 2$):

  There exists $k \in [\![1, \psi - 1]\!]$ such that $m = S^1_k$ and $m' = S^1_{k+1}$. If $k \leq \psi - 2$, then $\mathcal{G}(m') - \mathcal{G}(m) = b/r$ and $\text{size}(m') = b$ so the result holds. If $k = \psi - 1$, then

$$\begin{aligned}\mathcal{G}(m') - \mathcal{G}(m) &= d_2 - D_1 - \psi b/r \\ &= \text{size}(m')/r\end{aligned}$$

- Case $m \in S^1$, $m' \in X$: clear (Table B.4)
- Case $m, m' \in X$: clear as well (Table B.4) □

Lemma B.6 (The source described in the previous Subsection complies with the arrival curve $\gamma_{r,b}$)
The traffic generation described in the previous Subsection is $\gamma_{r,b}$-constrained.

Proof of Lemma B.6. Consider any set of n consecutive data-units generated by the source $(P_v)_{v\in[\![1,n]\!]}$. Then

$$\begin{aligned}\mathcal{G}(P_n) - \mathcal{G}(P_1) &\geq \sum_{v\in[\![1,n-1]\!]} \mathcal{G}(P_{v+1}) - \mathcal{G}(P_v) \\ &\geq \sum_{v\in[\![1,n-1]\!]} \frac{\text{size}(P_{v+1})}{r} \qquad \triangleright \text{ Lemma B.5} \\ r(\mathcal{G}(P_n) - \mathcal{G}(P_1)) + b &\geq \sum_{v\in[\![1,n-1]\!]} \text{size}(P_{v+1}) + b \\ &\geq \sum_{v\in[\![2,n]\!]} \text{size}(P_v) + b\end{aligned}$$

Table B.5: Absolute Release Time for Each Packet at the Output of Each System S_1, S_2

	Data unit, m	$\mathcal{G}(m)$, generation time	$\mathcal{E}_1(m)$, exit time out of S_1 for the packet transporting m through S_1	$\mathcal{E}_2(m)$, exit time out of S_2 for the packet transporting m through S_2
	I^2	0	$+\infty$	D_2
$\forall k \in [\![1, \chi^2 - 1]\!]$,	B^2_k	$k\frac{b}{r}$	$+\infty$	D_2
	$B^2_{\chi^2}$	$D_2 - d_2$	$+\infty$	D_2
$\forall k \in [\![1, \psi - 1]\!]$,	S^2_k	$D_2 - d_2 + k\frac{b}{r}$	$+\infty$	$D_2 + k\frac{b}{r}$
	S^2_ψ	$D_2 - D_1 - \frac{b}{r}$	$+\infty$	$D_2 + d_2 - D_1 - \frac{b}{r}$
	I^1	$D_2 - D_1$	D_2	$+\infty$
$\forall k \in [\![1, \chi^1 - 1]\!]$,	B^1_k	$D_2 - D_1 + k\frac{b}{r}$	D_2	$+\infty$
	$B^1_{\chi^1}$	$D_2 - d_1$	D_2	$+\infty$
$\forall k \in [\![1, \psi - 1]\!]$,	S^1_k	$D_2 - d_1 + k\frac{b}{r}$	$D_2 + k\frac{b}{r}$	$+\infty$
	S^1_ψ	$D_2 + d_2 - D_1 - d_1 - \frac{b}{r}$	$D_2 + d_2 - D_1 - \frac{b}{r}$	$+\infty$
$\forall k \in \mathbb{N}^*$,	X_k	$D_2 + d_2 - D_1 - d_1 + (k-1)\frac{b}{r}$	$D_2 + d_2 - D_1 + (k-1)\frac{b}{r}$	$+\infty$

Applying Lemma B.5, size$(P_1) \leq b$, so we obtain

$$r(\mathcal{G}(P_n) - \mathcal{G}(P_1)) + b \geq \sum_{v \in [\![1,n]\!]} \text{size}(P_v) \tag{B.15}$$

Equation (B.15) is the max-plus representation of a packetized flow constrained by an arrival curve $\gamma_{r,b}$ Le Boudec, Thiran 2001, §3. □

Description of the systems S_1, S_2

For a data unit m, we note $\mathcal{E}_1(m)$ [resp., $\mathcal{E}_2(m)$] the absolute time at which the packet transporting m through S_1 [resp., through S_2], exits S_1 [resp., exits S_2]. For $j \in \{1, 2\}$, we note $\mathcal{E}_j(m) = +\infty$ if and only if the packet transporting data unit m through S_j is lost by system S_j. Systems S_1 and S_2 release the packets generated by the source at the time instants shown in Table B.5.

Remark: We chose a system such that any packet transporting a data unit of category I^2, B^2 or S^2 is lost within S_1 ($\mathcal{E}_1(m) = +\infty$) and any packet transporting a data unit of category I^1, B^1, S^1 or X is lost within S_2 ($\mathcal{E}_2(m) = +\infty$). This scheme keeps the proof simple but note that a similar proof could be obtained assuming S_2 is lossless. In that case, we would only need to make sure that, for m in category I^1, B^1, S^1 or X, the packet transporting m through S_2 exits S_2 after the packet transporting m through S_1 exits S_1, i.e., *$\mathcal{E}_2(m) \geq \mathcal{E}_1(m)$ for any m.*

As an illustration, Figure B.8 shows the obtained cumulative function at the output of the PEF when applying the exit time instants of Table B.5 on the example of Figure B.7. Dashed boxes represent values of interest. We also plot on top of it the arrival curve at the output of the PEF, obtained in (B.10). In the following subsections, we prove that there exists no better VBR-arrival curve than this one for the shown output cumulative function.

Properties of the systems S_1, S_2

We now show the following properties of the above-described systems S_1, S_2.

Lemma B.7 (The delay bounds through S_1, S_2)
The delay of any non-lost packet through S_1 is bounded between d_1 and D_1. The delay of any non-lost packet through S_2 is bounded between d_2 and D_2.

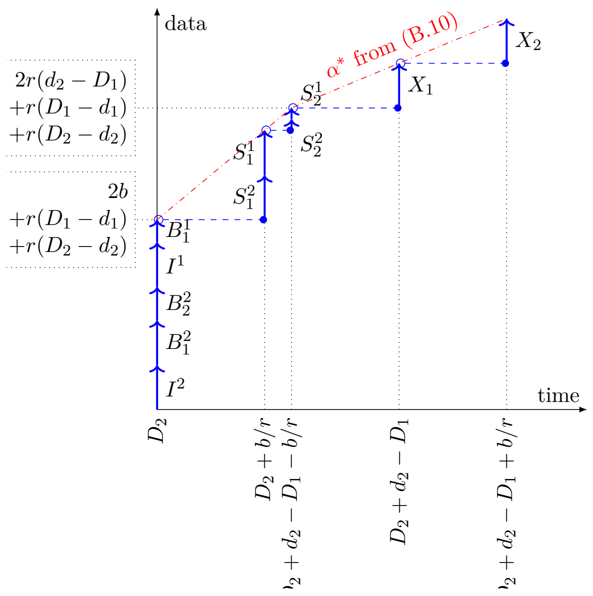


Figure B.8: Dashed blue: Cumulative function R^* at the output of the PEF, from the example trajectory of Figure B.7. Dashed boxes: Values of interest of the cumulative function. Red dashdotted: Arrival curve obtained from Corollary 4.1 and recalled in Equation (B.10).

Proof of Lemma B.7. From Table B.5, the result is clear for packets transporting data-units of categories I, S and X

We prove it for packets transporting data-units of category B:

Through S_1: for m a data unit of category B^2, the packet transporting m through S_1 is lost. For m a data unit of category B^1, the packet transporting m through S_1 verifies

$$\mathcal{G}(B^1_{\chi^1}) \geq \mathcal{G}(m) \geq \mathcal{G}(I^1)$$

because data units of B^1 are sent after I^1 and before $B^1_{\chi^1}$.

$$D_2 - (D_2 - d_1) \leq \mathcal{E}_1(m) - \mathcal{G}(m) \leq D_2 - (D_2 - D_1) \tag{B.16}$$

per Table B.5. Equation (B.16) proves that any packet transporting a data unit of type B^1 through S_1 has a delay through S_1 bounded in $[d_1, D_1]$.

Through S_2: for m a data unit of category B^1, the packet transporting m through S_2 is lost. For m a data unit of category B^2, the packet transporting m through S_2 verifies

$$\mathcal{G}(B^2_{\chi^2}) \geq \mathcal{G}(m) \geq \mathcal{G}(I^2)$$

because data units B^2 are sent after I^2 and before $B^2_{\chi^2}$.

$$D_2 - (D_2 - d_2) \leq \mathcal{E}_1(m) - \mathcal{G}(m) \leq D_2 - 0 \tag{B.17}$$

Equation (B.17) proves that any packet transporting a data unit of type B^2 through S_2 has a delay through S_2 bounded in $[d_2, D_2]$.

□

Properties of the output cumulative function

Call R^* the output cumulative function of the flow at the output of the PEF. Any data unit m is released as soon as the first packet containing m is received from either S_1 or S_2. Therefore, for any time instant t,

$$R^*(t) = \sum_{\left\{ m \,\middle|\, \begin{array}{c} \mathcal{E}_1(m) < t \\ \text{or } \mathcal{E}_2(m) < t \end{array} \right\}} \text{size}(m) \tag{B.18}$$

We apply Equation (B.18) to obtain the value of the cumulative function at several time-instants of interest. We start with $t = D_2$.

$$R^*(D_2) = \sum_{\left\{ m \,\middle|\, \begin{array}{c} \mathcal{E}_1(m) < D_2 \\ \text{or } \mathcal{E}_2(m) < D_2 \end{array} \right\}} \text{size}(m)$$

Per Table B.5, we obtain

$$R^*(D_2) = 0 \tag{B.19}$$

We then continue with $D_2 + \epsilon$ for $\epsilon > 0$,

$$\begin{aligned} & R^*(D_2 + \epsilon) \\ & = \sum_{\min(\mathcal{E}_1(m), \mathcal{E}_2(m)) < D_2 + \epsilon} \text{size}(m) \\ & \geq \sum_{m \in I} \text{size}(m) + \sum_{m \in B^1} \text{size}(m) + \sum_{m \in B^2} \text{size}(m) \\ & \triangleright \text{ From Table B.5} \end{aligned}$$

With (B.13), we obtain

$$\forall \epsilon > 0, \qquad R^*(D_2 + \epsilon) \geq 2b + r(D_1 - d_1) + r(D_2 - d_2) \tag{B.20}$$

And finally, $\forall \epsilon > 0$,

$$\begin{aligned} & R^*(D_2 + (d_2 - D_1) - b/r + \epsilon) \\ & = \sum_{\min(\mathcal{E}_1(m), \mathcal{E}_2(m)) < D_2 + (d_2 - D_1) - b/r + \epsilon} \text{size}(m) \\ & \geq \sum_{m \in I} \text{size}(m) + \sum_{m \in B} \text{size}(m) \\ & \quad + \sum_{m \in S_1} \text{size}(m) + \sum_{m \in S_2} \text{size}(m) \\ & \triangleright \text{ From Table B.5} \end{aligned}$$

With (B.14), we obtain $\forall \epsilon > 0$

$$\begin{aligned} &R^*(D_2 + (d_2 - D_1) - b/r + \epsilon) \\ &\geq r(D_1 - d_1) + r(D_2 - d_2) + 2r(d_2 - D_1) \end{aligned} \tag{B.21}$$

Properties of any candidate arrival curve for f

Consider any VBR arrival curve $\alpha' = \text{SPEC}(M', p', r', b')$ defined in Le Boudec, Thiran 2001, §1.2 and assume that α' is an arrival curve for f at the output of the PEF.

Consider also the piecewise-linear function $\alpha^\dagger$ defined on $\mathbb{R}+$ by

$$\alpha^\dagger : t \mapsto \min(M' + \rho' t, b' + r' t) \tag{B.22}$$

By definition, we have, for all $t \geq 0$

$$\alpha'(t) = \begin{cases} \alpha^\dagger(t) & \text{if } t > 0 \\ 0 & \text{if } t = 0 \end{cases} \tag{B.23}$$

Note that $\alpha^\dagger$ is concave and wide-sense increasing. We also have the following result

Lemma B.8
For any $s \geq t \geq 0$, $\alpha^\dagger(s) - \alpha^\dagger(t) \geq r'(s - t)$

Proof of Lemma B.8. We simply break in all the possible cases: • If both $s \leq \frac{M'-b'}{\rho'-r'}$ and $t \leq \frac{M'-b'}{\rho'-r'}$ then $\alpha^\dagger(s) - \alpha^\dagger(t) = \rho'(s - t) \geq r'(s - t)$ because $r' \geq \rho'$.

• If $t \leq \frac{M'-b'}{\rho'-r'}$, and $s \geq \frac{M'-b'}{\rho'-r'}$, then $\alpha^\dagger(s) - \alpha^\dagger(t) = b' - M' + r's - \rho' t \geq r's - \rho' t \geq r'(s-t)$ because $b' \geq M'$ and $\rho' \geq r'$.

• If both $s \geq \frac{M'-b'}{\rho'-r'}$ and $t \geq \frac{M'-b'}{\rho'-r'}$ then $\alpha^\dagger(s) - \alpha^\dagger(t) = r'(s - t)$. □

We observe that after $D_2 + (d_2 - D_1)$, the output traffic R^* is made of the data units of category X with a size b and a period b/r. Therefore, the long-term rate of the flow at the output of the PEF is exactly r and any piece-wise linear arrival curve for this flow must have a long-term rate at least as big as r. For the VBR arrival curve α', this gives

$$r' \geq r \tag{B.24}$$

Then, as α' is an arrival curve for f at the output of the PEF, by Definition 2.5, for any $t, s \geq 0$, $R^*(t + s) - R(t) \leq \alpha'(s)$.

In particular, $\forall \epsilon > 0$

$$\alpha'(\epsilon) \geq R^*(D_2 + \epsilon) - R^*(D_2)$$

With (B.19) and (B.20) this gives, $\forall \epsilon > 0$,

$$\alpha'(\epsilon) \geq 2b + r(D_1 - d_1 + D_2 - d_2) \tag{B.25}$$

This is valid for any choice of $\epsilon > 0$ thus $\lim_{t\to 0} \alpha'(t) \geq 2b + r(D_1 - d_1 + D_2 - d_2)$, which gives:

$$\alpha^{\dagger}(0) \geq 2b + r(D_1 - d_1 + D_2 - d_2) \tag{B.26}$$

Similarly, $\forall \epsilon > 0$,

$$\alpha'\left(d_2 - D_1 - \frac{b}{r} + \epsilon\right) \geq R^*\left(D_2 + d_2 - D_1 - \frac{b}{r} + \epsilon\right) - R^*(D_2) \tag{B.27}$$

with (B.19) and (B.21), we obtain, $\forall \epsilon > 0$,

$$\alpha'\left(d_2 - D_1 - \frac{b}{r} + \epsilon\right) \geq r(D_1 - d_1) + r(D_2 - d_2) + 2r(d_2 - D_1) \tag{B.28}$$

this is again valid for any value $\epsilon > 0$ so $\lim_{\epsilon \to 0} \alpha'(d_2 - D_1 - \frac{b}{r} + \epsilon) \geq r(D_1 - d_1) + r(D_2 - d_2) + 2r(d_2 - D_1)$, which gives

$$\begin{aligned} \alpha^{\dagger}\left(d_2 - D_1 - \frac{b}{r}\right) &= \alpha'\left(d_2 - D_1 - \frac{b}{r}\right) \\ &\geq r(D_1 - d_1) + r(D_2 - d_2) + 2r(d_2 - D_1) \end{aligned} \tag{B.29}$$

And using the above properties, we can prove the following result

Lemma B.9
For any $t > 0$, $\alpha^{\dagger}(t) \geq \alpha^*(t)$ where α^* is the VBR obtained by applying Corollary 4.1 and given in (B.10).

Proof of Lemma B.9. • If $t > d_2 - D_1 - \frac{b}{r}$, then

$$\begin{aligned} \alpha^{\dagger}(t) &= \alpha^{\dagger}\left(d_2 - D_1 - \frac{b}{r}\right) + \alpha^{\dagger}(t) - \alpha^{\dagger}\left(d_2 - D_1 - \frac{b}{r}\right) \\ &\geq \alpha^{\dagger}\left(d_2 - D_1 - \frac{b}{r}\right) + r'(t - d_2 + D_1) + b && \triangleright \text{ Lemma B.8} \\ &\geq \alpha^{\dagger}\left(d_2 - D_1 - \frac{b}{r}\right) + r(t - d_2 + D_1) + b && \triangleright \text{ (B.24)} \\ &\geq r(D_1 - d_1) + r(D_2 - d_2) + 2r(d_2 - D_1) + r(t - d_2 + D_1) + b && \triangleright \text{ (B.29)} \\ &\geq b + rD_2 - rd_1 + rt = \alpha^*(t) && \triangleright \text{ (B.10)} \end{aligned}$$

• If $0 < t < d_2 - D_1 - \frac{b}{r}$, then we use the fact that $\alpha^{\dagger}$ is concave on $\mathcal{R}+$.

Define

$$x = \frac{t}{d_2 - D_1 - \frac{b}{r}} \tag{B.30}$$

then, by definition of a concave function,

$$\begin{aligned}
\alpha^\dagger(t) &\geq x\alpha^\dagger\left(d_2 - D_1 - \frac{b}{r}\right) + (1-x)\alpha^\dagger(0) \\
&\geq x\left(r(D_1-d_1) + r(D_2-d_2) + 2r(d_2-D_1)\right) && \triangleright\ (\text{B.29}) \\
&\quad + (1-x)\left(2b + r(D_1 - d_1 + D_2 - d_2)\right) && \triangleright\ (\text{B.26}) \\
&\geq 2rx\left(d_2 - D_1 - \frac{b}{r}\right) + 2b + r(D_1-d_1) + r(D_2-d_2) \\
&\geq 2rt + 2b + r(D_1-d_1) + r(D_2-d_2) = \alpha^*(t) && \triangleright\ (\text{B.26}) \text{ and } (\text{B.10})
\end{aligned}$$

□

By (B.23), Lemma B.9 proves that $\forall t > 0$, $\alpha'(t) \geq \alpha^*(t)$ and by definition of an arrival curve, $\alpha'(0) = \alpha^*(0) = 0$. We hence have proved that α^* is a better arrival curve for f at the output of the PEF than α'. This is valid for any VBR curve α' that is an arrival curve for f at the output of the PEF. Therefore α^* obtained using Corollary 4.1 is the best VBR arrival curve for f at the output of the PEF.

Case $d_2 - D_1 \leq b/r$:

In this case, the leaky-bucket $\gamma_{2r,2b+r(D_1-d_1+D_2-d_2)}$ is always larger than $\gamma_{r,b+r(D_2-d_1)}$. Thus the application of Corollary 4.1 gives that the leaky-bucket $\alpha^* = \gamma_{r,b+r(D_2-d_1)}$ is an arrival curve for f at the output of the PEF in Figure 4.13.

Using the same rationale as for the previous case, we can use a greedy source that generates packets with a long-term rate of r, thus any arrival curve for f at the output of the PEF must also have a long-term rate larger than r.

Therefore, the proof of tightness needs only to exhibit a trajectory that creates a burst as big as $b + r(D_2 - d_1)$. This is done as follows.

Definition of several constants

We define

$$\chi^1 \triangleq \left\lceil \frac{r(D_1-d_1)}{b} \right\rceil \quad \text{and} \quad \chi^2 \triangleq \left\lceil \frac{r(D_2-D_1)}{b} \right\rceil \tag{B.31}$$

Note that both $\chi^1 \geq 1$ and $\chi^2 \geq 1$. We further consider a time instant t_0 such that $t_0 > D_2 - D_1$.

Description of the traffic generation at the source

We classify the data units generated by the source into two categories: I, B. Category B is then subdivided into subcategories B^1 and B^2. The category of a data unit defines the role that the data unit has in the trajectory. This notion is only used in the proof and does not relate to any physical property of the data units.

Category I The source generates a unique data unit I at absolute time t_0, of length b (see Table B.6).

Table B.6: Generation of the Data-Unit of Category I in the Trajectory that Achieves the Tightness of Corollary 4.1, when $d_2 - D_1 \leq b/r$.

Data unit m	Size, size(m)	Generation time, $\mathcal{G}(m)$
I	b	t_0

Table B.7: Generation of the Data-Units of Category B in the Trajectory that Achieves the Tightness of Corollary 4.1, when $d_2 - D_1 \leq b/r$.

Data unit m		Size, size(m)	Generation time, $\mathcal{G}(m)$
$\forall k \in [\![1, \chi^2 - 1]\!]$,	B^2_k	b	$t_0 - D_2 + D_1 + (k-1)\frac{b}{r}$
	$B^2_{\chi^2}$	$r(D_2 - D_1) - (\chi^2 - 1)b$	$t_0 - b/r$
$\forall k \in [\![1, \chi^1 - 1]\!]$,	B^1_k	b	$t_0 + k\frac{b}{r}$
	$B^1_{\chi^1}$	$r(D_1 - d_1) - (\chi^1 - 1)b$	$t_0 + (D_1 - d_1)$

Note: The role of the data unit I is to create the term b of the burst $b + r(D_2 - d_1)$.

Category B In addition, the source generates χ^1 data units of subcategory B^1 and χ^2 data units of subcategory B^2, as described in Table B.7.

A possible output of the source when combining categories I and B is shown in Figure B.9. In the proposed situation, we have $\chi^1 = \chi^2 = 2$. Both subcategories B^1 and B^2 are made of two data units. The data units of B^2 are sent before I whereas the data units of B^1 are sent after I.

We note that, for any value of χ^1, χ^2,

$$\sum_{k \in [\![1,\chi^1]\!]} \text{size}(B^1_k) = r(D_1 - d_1) \quad \text{and} \quad \sum_{k \in [\![1,\chi^2]\!]} \text{size}(B^2_k) = r(D_2 - D_1) \tag{B.32}$$

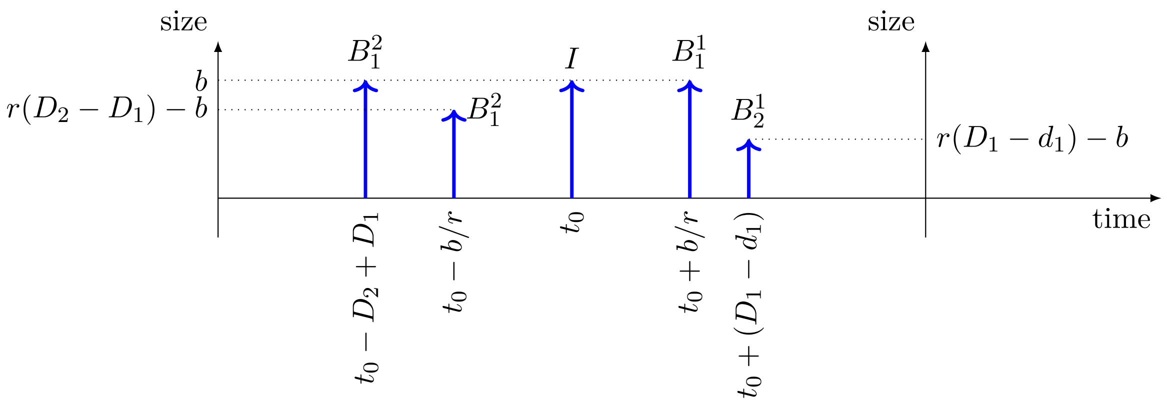


Figure B.9: Example of the source output in the trajectory that achieves the tightness, focusing on categories I and B.

Table B.8: Absolute Release Time for Each Packet at the Output of Each System S_1, S_2, in the Case $d_2 - D_1 \leq b/r$.

	Data unit, m	$\mathcal{G}(m)$, Generation time	$\mathcal{E}_1(m)$, exit time out of S_1 for the packet transporting m through S_1	$\mathcal{E}_2(m)$, exit time out of S_2 for the packet transporting m through S_2
$\forall k \in [\![1, \chi^2 - 1]\!]$,	B^2_k	$t_0 - D_2 + D_1 + (k-1)\frac{b}{r}$	$+\infty$	$t_0 + D_1$
	$B^2_{\chi^2}$	$t_0 - b/r$	$+\infty$	$t_0 + D_1$
	I	t_0	$t_0 + D_1$	$+\infty$
$\forall k \in [\![1, \chi^1 - 1]\!]$,	B^1_k	$t_0 + k\frac{b}{r}$	$t_0 + D_1$	$+\infty$
	$B^1_{\chi^1}$	$t_0 + (D_1 - d_1)$	$t_0 + D_1$	$+\infty$

Properties of the traffic generation at the source

As for the previous case, we prove that Lemmas B.4 and B.5 hold for the traffic described in B.2.4. This is clear for most pair of data units, let us show it for example for data units $B^2_{\chi^2-1}$ and $B^2_{\chi^2}$:

$$\begin{aligned}\mathcal{G}(B^2_{\chi^2}) - \mathcal{G}(B^2_{\chi^2-1}) &= t_0 - \frac{b}{r} - t_0 + D_2 - D_1 - (\chi^2 - 1)\frac{b}{r} + \frac{b}{r}\\ &= D_2 - D_1 - (\chi^2-1)\frac{b}{r}\\ &\geq \frac{\mathrm{size}(B^2_{\chi^2})}{r}\end{aligned}$$

Therefore, Lemma B.6 also holds for the traffic described in B.2.4. The traffic described in the trajectory is $\gamma_{r,b}$-constrained.

Description of the systems S_1, S_2

With the same notations and conventions as for the previous case, the systems S_1 and S_2 release the packets containing the different data units at the time instants shown in Table B.8.

Properties of the systems S_1, S_2

As for the previous case, we can also prove that Lemma B.7 holds for the systems S_1, S_2 described above. This is clear from Table B.8 for most data units. For example, the delay of the packet transporting the data unit $B^2_{\chi^2}$ through S_2 is at least d_2 because, by assumption, $d_2 - D_1 \leq \frac{b}{r}$.

Properties of the output cumulative function

In the trajectory of Table B.8, all data units exit the PEF at $t_0 + D_1$. We hence have created a burst of size

$$\begin{aligned}&\mathrm{size}(I) + \sum_{k\in[\![1,\chi^1]\!]} \mathrm{size}(B^1_k) + \sum_{k\in[\![1,\chi^2]\!]} \mathrm{size}(B^2_k)\\ &= b + r(D_1 - d_1) + r(D_2 - D_1) && \triangleright \text{(B.32)}\\ &= b + r(D_2 - d_1)\end{aligned}$$

Therefore, any curve that is an arrival curve of f at the output of the PEF should have a limit at 0 at least larger than $b + r(D_2 - d_1)$ and a long-term rate at least larger than r. Thus any such curve that is also concave on $\mathbb{R}^*+$ must hence be larger than the leaky-bucket arrival curve $\gamma_{r,b+r(D_2-d_1)}$. In particular, any VBR arrival curve (concave on $\mathbb{R}^*+$ by definition) is larger than $\gamma_{r,b+r(D_2-d_1)}$.

□

B.2.5 Proof of Proposition 4.2

Proof of Proposition 4.2. Consider the flow f and two observation points v, w that meet the constraints of Definition 4.5: v is in vertex n, w is in vertex o, n is not an EP-vertex of $\mathcal{G}(f)$, o is a diamond ancestor of n in $\mathcal{G}(f)$, and the flow f is packetized at v, w. Define m_k, l_k, E_k as in Definition 4.5.

E_k is correctly defined because n is not an EP-vertex, thus m_k can cross the observation point v at most once. Furthermore, f is packetized at v, thus all the bits of m_k cross v at the same time. Define the reordering offset of m_k [Mohammadpour, Le Boudec 2021, Eq. (4)] by

$$\Pi_k \triangleq \sum_{j|j>k, E_j<E_k} l_j \tag{B.33}$$

Denote by $R_{f,v}$ the cumulative arrival function of flow f at observation point v. From Definition 2.4, $R_{f,v}(t)$ is the number of bits of flow f that cross v over the time interval $[0, t]$. The cumulative functions are assumed left-continuous (Section 2.2), thus $R_{f,v}(t)$ is also number of bits of flow f that cross v over the time interval $[0, t[$. Hence, for any non-lost data unit m_k, $R_{f,v}(E_k)$ is the number of bits of f that cross v strictly before E_k. As f is packetized at v, $R_{f,v}(E_k)$ is hence the sum of the length of the packet lengths for all data units that arrived before m_k, excepted m_k.

Hence Π_k can be written

$$\begin{aligned}
\Pi_k &= \sum_{j|j>k, E_j<E_k} l_j \\
&= R_{f,v}(E_k) - R_{f,v}(\min_{j>k} E_j) \\
&\leq \alpha_{f,v}(E_k - \min_{j>k} E_j)
\end{aligned}$$

because $\alpha_{f,v}$ is an arrival curve of f at v. Denote by $\Lambda_k \triangleq E_k - \min_{j>k} E_j$ the reordering late offset of data unit k [Mohammadpour, Le Boudec 2021, Eq. (2)]. We hence obtain that for all k such that $E_k < +\infty$,

$$\Pi_k \leq \alpha_{f,v}(\Lambda_k) \tag{B.34}$$

The RTO and the RBO of f at v with respect to w are defined in Definition 4.5 by

$$\pi_{f,v}(w) \triangleq \sup_{k|E_k<+\infty} \Pi_k \qquad \lambda_{f,v}(w) \triangleq \sup_{k|E_k<+\infty} \Lambda_k \tag{B.35}$$

Equation (B.34) is valid for any k such that $E_k < +\infty$; $\alpha_{f,v}$ is a wide-sense increasing function; and $\sup_{k|E_k<+\infty} \Lambda_k$ is bounded by assumption. We hence obtain $\pi_{f,v}(w) \leq \alpha_{f,v}(\lambda_{f,v}(w))$.

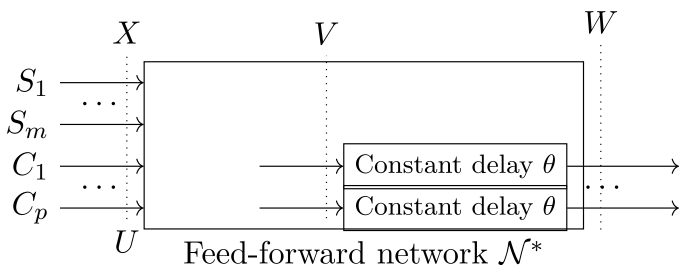


Figure B.10: Unfolding $\mathcal{N}$ into a feed-forward causal system.

□

B.2.6 Proof of Theorem 4.2

Proof of Theorem 4.2. The section of the network between the diamond ancestor a and the vertex n that contains the PEF is a system (neither FIFO nor lossless in general) with a jitter for f bounded by $D_f^{a^* \to n^{\text{in}}} - d_f^{a^* \to n^{\text{in}}}$. We apply [Mohammadpour, Le Boudec 2021, Thm 5] to obtain the result. □

B.2.7 Proof of Theorem 4.3

Proof of Theorem 4.3. Consider network $\mathcal{N}$ with cyclic dependencies and select θ such that $0 < \theta < T^{\min}$ where $T^{\min}$ is the minimum propagation time of the transmission links (see Chapter 2, Section 2.6). Let $\mathcal{E}$ be a feedback arc set for the network's underlying graph $\mathcal{G}(\mathcal{N})$. For each edge e in $\mathcal{E}$, we define one cut per flow going through e, and we consider $\{C_i\}_i$ the set of these cuts. In other words, $\{C_i\}_i = \{(e, f); \forall e \in \mathcal{E}; \forall f | e \in \text{edges}(\mathcal{G}(f))\}$. We consider also $\{S_j\}_j$ the set of the sources in the network $\mathcal{N}$.

Using these notations, we unfold the network $\mathcal{N}$ as in Figure B.10.

- W [resp. U] represent the points located just before [resp. just after] the $\{C_i\}_i$ cuts: in $\mathcal{N}$, W and U are connected together through links not shown here : $\forall i; W_i \to U_i$.
- X represents the points located just after the sources $\{S_j\}_j$.
- Last, V represents the points located exactly θ seconds before W. For each i an index of W, the point V_i is located on the same link as W_i because t_{prop} is the minimum propagation delay.

We call $\mathcal{N}^*$ the system between (U, X) and W. $\mathcal{N}^*$ is a causal system as the sources have been extracted. It is also feed-forward because $\mathcal{E}$ is a feedback arc set. Since $\mathcal{N}$ is not assumed to be FIFO nor lossless, neither is $\mathcal{N}^*$. Call $\mathcal{FF} : (\mathbf{b}_{\text{in}}, \mathbf{d}_{\text{in}}) \mapsto (\mathbf{b}_{\text{out}}, \mathbf{d}_{\text{out}})$ the feed-forward algorithm that computes;

- the burst bounds $\mathbf{b}_{\text{out}}$, and
- the upper delay bounds with respect to the flow's source $\mathbf{d}_{\text{out}}$

Table B.9: Notations for the Proof of Theorem 4.3

	Vectors of observation points located respectively
X	at the output of the sources,
U	just after the cuts,
W	just before the cuts and
V	θ seconds before W.
$\mathcal{T}$	An acceptable trajectory for the network
$\mathcal{T}(\tau)$	The trajectory that is identical to $\mathcal{T}$ before τ
	and drops all packets at X and U after τ.
	Cumulative process at M_i, respectively
$R_{M,i}(t)$	on trajectory $\mathcal{T}$,
$R^{\tau}_{M,i}(t)$	on trajectory $\mathcal{T}$ when the observation is stopped at τ and
$R'^{\tau}_{M,i}(t)$	when the input is stopped at τ (trajectory $\mathcal{T}(\tau)$)
	The sequence of the measured source-to-M delays
	for the data units that cross M_i, respectively
$(D_{M,i})$	on trajectory $\mathcal{T}$,
$(D^{\tau}_{M,i})$	on trajectory $\mathcal{T}$ when the observation is stopped at τ and
$(D'^{\tau}_{M,i})$	when the input is stopped at τ (trajectory $\mathcal{T}(\tau)$)

at the output of $\mathcal{N}^*$ based on the equally defined bounds $(\mathbf{b}_{\text{in}}, \mathbf{d}_{\text{in}})$ provided at the input of $\mathcal{N}^*$.

We consider now a non-negative fixed-point $(\overline{\mathbf{b}}, \overline{\mathbf{d}})$, *i.e.* such that $\mathcal{FF}(\overline{\mathbf{b}}, \overline{\mathbf{d}}) = (\overline{\mathbf{b}}, \overline{\mathbf{d}})$. Our goal is to prove that the elements of the vectors $(\overline{\mathbf{b}}, \overline{\mathbf{d}})$ constitute valid burst and delay bounds at the cuts. The notations used in this proof are regrouped in Table B.9.

Consider any acceptable trajectory $\mathcal{T}$ for $\mathcal{N}$. To ease the notation, the dependency of the following notions on $\mathcal{T}$ is made implicit.

Cumulative processes and true delays. For $M \in \{X, U, V, W\}$, we note $\mathbf{R}_M(t) = [R_{M,i}(t)]_i$ the vector of the cumulative processes in this trajectory, observed at points M_i.

For $M \in \{X, U, V, W\}$ and i and index of M, we note $(D_{M,i,n})_{n\in\mathbb{N}^*}$ the sequence of the measured source-to-M delays for the data units that cross M_i, in the order at which they present themselves at M_i. If the process at M_i is not packetized, then the instant at which a data unit crosses M_i is defined as the instant of the transmission of the last bit of the data unit. If several packets arrive at the same instant at the observation point M_i, then an arbitrary deterministic second order (for example, based on the packets' content) is used to break ties. If a same data unit presents itself twice at M_i (for example when it is transported by two redundant packets), then each traversal generates a distinct entry in the sequence $(D_{M,i,n})_{n\in\mathbb{N}^*}$.

Example: Call f the flow cut by C_i. If P is the third packet of f observed at M_i, then $D_{M,i,3}$ is the time elapsed between the emission of the data unit contained in P at the source of f and the the instant a which P goes through M_i.

We note $\mathbf{D}_M = [(D_{M,i})]_i$ the vector of all such sequences at the observation point M.

In the rest of the proof, we take any $\tau \geq 0$.

Stopping the observation at τ. For $M \in \{U, V, W\}$ we call $\mathbf{R}_M^\tau(t)$ the vector of the cumulative processes stopped at τ: $\mathbf{R}_M^\tau : t \mapsto \mathbf{min}(\mathbf{R}_M(t), \mathbf{R}_M(t)) = [\min(R_{M,i}(t), R_{M,i}(\tau))]_i$.

Similarly, we call $\mathbf{D}_M^\tau$ the vector of the sequences at M obtained from the sequences $\mathbf{D}_M$ by keeping only the delays relative to the packets that arrive at M strictly before the absolute time reaches τ.

Lemma B.10 ($\mathbf{D}_M^\tau$ is a finite prefix of $\mathbf{D}_M$)
For $M \in \{U, V, W\}$, for i an index of M, there exists $n_{last} \geq 0$ such that $(D_{M,i}^\tau) = (D_{M,i,n})_{n \leq n_{last}}$. If $n_{last} = 0$, then $(D_{M,i}^\tau)$ is an empty sequence.

Proof. Call f the unique flow cut by C_i (by definition of the cuts). If no packet has been observed before τ, then we set $n_{\text{last}} = 0$, and the result holds. If one or more packets of f are observed before τ at M_i, we select n_{last} to be the index within the sequence $(D_{M,i})$ of the source-to-M delay that was measured for the last packet that arrived at M_i strictly before τ. Between 0 and τ, a finite duration, the source of flow f could not have produced an infinite number of packets (by assumption on the sources). $\mathcal{N}^*$ is causal and the number of nodes in the network is finite. In addition, flow graphs are acyclic, hence the number of packets of f crossing M_i between 0 and τ is also finite, so n_{last} is finite. Any packet that arrived strictly before τ has arrived before the last packet that arrived strictly before τ, hence the result holds by definition of $D_{M,i}^\tau$. □

Stopping the input at τ. Recall that the above definitions depend on the trajectory $\mathcal{T}$. We define $\mathcal{T}'(\tau)$ the trajectory obtained from $\mathcal{T}$ by stopping all the processes at U and X at τ: in trajectory $\mathcal{T}'(\tau)$, for $t \geq \tau$, all the data, all the packets that go through X and U are dropped (deleted).

We call $\mathbf{R}_M'^\tau(t)$ [resp. $\mathbf{D}_M'^\tau$] the version of $\mathbf{R}_M(t)$ [resp. $\mathbf{D}_M$] when defined on the trajectory $\mathcal{T}'(\tau)$ instead of being defined for trajectory $\mathcal{T}$.

Lemma B.11
For $M \in \{U, V, W\}$, for i an index of M,

1. *$D_{M,i}^\tau$ is a prefix of $D_{M,i}'^\tau$.*

2. *$D_{M,i}'^\tau$ is a finite sequence.*

Proof.

1. $D_{M,i}^\tau$ is a finite sequence that contains all the source-to-M delay measured for packets reaching M_i before τ. Before τ, the trajectories $\mathcal{T}$ and $\mathcal{T}(\tau)$ are identical (same processes, packets in the same order), hence $D_{M,i}^\tau$ is contained at the beginning of $D_{M,i}'^\tau$, it is a prefix.

2. Since $\mathcal{N}^*$ is causal, after τ, no more packet is produced in or enters into $\mathcal{N}^*$. Since the number of nodes in the network is finite and since flow graphs are acyclic, the observation point M_i can only observe the data unit remaining in the network a finite number of times, hence $D_{M,i}'^\tau$ is finite.

□

Obtaining the worst-case bounds. For $M \in \{U, V, W\}$ we denote by $\mathbf{b}^{\tau}_{M}$ [resp. $\mathbf{b}'^{\tau}_{M}$] the vector of worst-case leaky-bucket bursts observed at M on the observation stopped at τ [resp. on the trajectory $\mathcal{T}(\tau)$]. By definition, they correspond to the worst-case leaky-bucket bursts observed on cumulative processes $\mathbf{R}^{\tau}_{M}(t)$ [resp. $\mathbf{R}'^{\tau}_{M}(t)$]. For example, for any i an index of M, $b^{\tau}_{M,i} = \sup_{t' \geq t}(R^{\tau}_{M,i}(t') - R^{\tau}_{M,i}(t) - r(t'-t))$ where r is the leaky-bucket rate of the flow cut by C_i.

Similarly, for $M \in \{U, V, W\}$, we denote by $\mathbf{d}^{\tau}_{M}$ [resp. $\mathbf{d}'^{\tau}_{M}$] the worst-case source-to-M delay observed at M on the observation stopped at τ [resp. on the trajectory $\mathcal{T}(\tau)$]. By Lemma B.10 and B.11, sequences $\mathbf{D}^{\tau}_{M}$ and $\mathbf{D}'^{\tau}_{M}$ are either empty or finite. We take the convention that the maximum of an empty sequence is 0 seconds of true delay. Hence, $\mathbf{d}^{\tau}_{M}$ [resp. $\mathbf{d}'^{\tau}_{M}$] correspond by definition to the maximum value of the sequence $\mathbf{D}^{\tau}_{M}$ [resp. $\mathbf{D}'^{\tau}_{M}$]. For example, for i an index of M, call n_{last} the last index of the finite sequence $D^{\tau}_{M,i}$. Then $d^{\tau}_{M,i} = \max_{j \in [\![1, n_{\text{last}}]\!]} D^{\tau}_{M,i,j}$.

Lemma B.12
Focusing on $M = V$,

1. $\mathbf{b}^{\tau}_{V} \leq \mathbf{b}'^{\tau}_{V}$
2. $\mathbf{d}^{\tau}_{V} \leq \mathbf{d}'^{\tau}_{V}$

Proof.
1. Same proof as in B.1
2. For i an index of V, by Lemma B.11, $(D^{\tau}_{M,i})$ is a finite prefix of the finite sequence $(D'^{\tau}_{M,i})$, hence all values in $(D^{\tau}_{M,i})$ are contained in $(D'^{\tau}_{M,i})$, hence $\max_j(D^{\tau}_{M,i,j}) \leq \max_k(D'^{\tau}_{M,i,k})$

□

Using the $\mathcal{FF}$ bound. $\mathcal{FF}$ computes, for a given input, a bound on the bursts and the source-to-W true delay bounds for the non-lost packets at the output W. Hence $(\mathbf{b}'^{\tau}_{W}, \mathbf{d}'^{\tau}_{W}) \leq \mathcal{FF}(\mathbf{b}^{\tau}_{U}, \mathbf{d}^{\tau}_{U})$. The delay between V and W is constant equal to θ, hence for any index i of V, $\forall t \geq 0, \mathbf{R}'^{\tau}_{V,i}(t) = \mathbf{R}'^{\tau}_{W,i}(t+\theta)$, which, per the definition of $\mathbf{b}'^{\tau}_{M}$, gives $\mathbf{b}'^{\tau}_{V} = \mathbf{b}'^{\tau}_{W}$.

Similarly, all the packets that reached V_i in the trajectory $\mathcal{T}(\tau)$ will reach W_i exactly θ seconds after and in the same order because the link between V_i and W_i is a lossless constant-delay link. This gives $\mathbf{d}'^{\tau}_{V} = \mathbf{d}'^{\tau}_{W} - \boldsymbol{\theta}$ where $\boldsymbol{\theta}$ is a vector of the same size of V that contains θ in every field. It gives

$$(\mathbf{b}^{\tau}_{V}, \mathbf{d}^{\tau}_{V}) \leq \mathcal{FF}(\mathbf{b}^{\tau}_{U}, \mathbf{d}^{\tau}_{U}) - (\mathbf{0}, \theta)$$

Since the link between V and W is lossless with a constant delay equal to θ, W observes between θ and $\tau + \theta$ the same exact traffic than what V observes between 0 and τ, with the exception that the packets have θ more seconds of delay from their source. Hence $\mathbf{b}^{\tau+\theta}_{W} = \mathbf{b}^{\tau}_{V}$ and $\mathbf{d}^{\tau+\theta}_{W} = \mathbf{d}^{\tau+\theta}_{V} + \boldsymbol{\theta}$. Finally, W and U are connected together in $\mathcal{N}$, so we obtain

$$(\mathbf{b}^{\tau+\theta}_{U}, \mathbf{d}^{\tau+\theta}_{U}) = (\mathbf{b}^{\tau+\theta}_{W}, \mathbf{d}^{\tau+\theta}_{W}) = (\mathbf{b}^{\tau}_{V}, \mathbf{d}^{\tau}_{V}) + (\mathbf{0}, \boldsymbol{\theta}) \leq \mathcal{FF}(\mathbf{b}^{\tau}_{U}, \mathbf{d}^{\tau}_{U}) \tag{B.36}$$

Equation (B.36) is valid for any $\tau \geq 0$, so applying it with $\tau = k\theta, k \in \mathbb{N}$ gives

$$\forall k \in \mathbb{N}, \quad \left(\mathbf{b}^{(k+1)\theta}_{U}, \mathbf{d}^{(k+1)\theta}_{U}\right) \leq \mathcal{FF}(\mathbf{b}^{k\theta}_{U}, \mathbf{d}^{k\theta}_{U}) \tag{B.37}$$

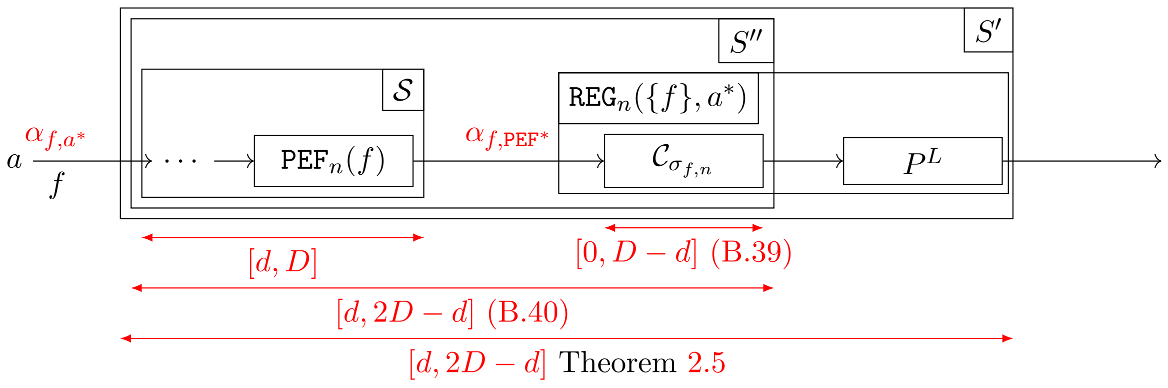


Figure B.11: Notations for the Proof of Theorem 4.4.

The network is empty at $t = 0$ so $(\mathbf{b}_U^0, \mathbf{d}_U^0) = (\mathbf{0}, \mathbf{0})$. $\mathcal{FF}$ can be assumed to be wide-sense increasing as per [Bouillard, Boyer, Le Corronc 2018, Chap. 12]. Combined with the fact that $\mathcal{FF}(\overline{\mathbf{b}}, \overline{\mathbf{d}}) = (\overline{\mathbf{b}}, \overline{\mathbf{d}})$ and a simple induction argument, it follows that $\left(\mathbf{b}_U^{k\theta}, \mathbf{d}_U^{k\theta}\right) \leq (\overline{\mathbf{b}}, \overline{\mathbf{d}})$ for all $k \in \mathbb{N}$. Last any $\tau > 0$, $(\mathbf{b}_U^{\tau}, \mathbf{d}_U^{\tau}) \leq \left(\mathbf{b}_U^{k\theta}, \mathbf{d}_U^{k\theta}\right)$ with $k = \lceil \frac{\tau}{\theta} \rceil$, so $\forall \tau > 0$, $(\mathbf{b}_U^{\tau}, \mathbf{d}_U^{\tau}) \leq (\overline{\mathbf{b}}, \overline{\mathbf{d}})$. Now the worst case bursts and virtual delays on this trajectory $\mathcal{T}$ are

$$(\mathbf{b}_U, \mathbf{d}_U) = \lim_{\tau \geq 0} (\mathbf{b}_U^{\tau}, \mathbf{d}_U^{\tau}) \leq (\overline{\mathbf{b}}, \overline{\mathbf{d}}) \tag{B.38}$$

Equation (B.38) is valid for any acceptable trajectory $\mathcal{T}$ of the network $\mathcal{N}$, so the network $\mathcal{N}$ is stable and $(\overline{\mathbf{b}}, \overline{\mathbf{d}})$ is a finite bound for $(\mathbf{b}_U, \mathbf{d}_U)_{\forall \mathcal{T}}$, the worst-case burst and delay bounds in the network $\mathcal{N}$. □

B.2.8 Proof of Theorem 4.4

Proof of Theorem 4.4. Consider the system $\mathcal{S}$ between the output of a and the output of the PEF (Figure B.11). We apply Item 2/ of Theorem 4.1 with diamond ancestor a. We obtain that

$$\begin{aligned}\alpha_{f,\texttt{PEF}^*} &= \alpha_{f,a^*} \oslash \delta_{D-d} \\ &= \gamma_{r,b} \oslash \delta_{D-d} \\ &= \gamma_{r,b+r(D-d)}\end{aligned}$$

is an arrival curve for f at the output of the PEF, *i.e.*, at the output of $\mathcal{S}$.

The input of the PFR is packetized because the PFR is located within a device, after the packetizer. In addition, $\sigma_{f,n} = \gamma_{r,b}$ is sub-additive and b is larger than the maximum packet length of f because $\gamma_{r,n}$ is an arrival curve for f at a^*. Hence, we can apply the service-curve characterization of the PFR (Proposition 3.1): the PFR with shaping curve $\sigma_{f,n}$ is realized by the concatenation of a greedy-shaper (Definition 2.7) with shaping curve $\sigma_{f,n}$ followed by a packetizer.

As $\sigma_{f,n} = \gamma_{r,b}$, it is sub-additive, we apply the first item of Theorem 2.4: the greedy shaper $\mathcal{C}_{\sigma_{f,n}}$ offers to f the service-curve $\sigma_{f,n}$. We apply Theorem 2.2 and obtain that the horizontal deviation

$$\begin{aligned}&\mathfrak{h}(\alpha_{f,\mathtt{PEF}^*}, \sigma_{f,n})\\ =&\mathfrak{h}(\gamma_{r,b+r(D-d)}, \gamma_{r,b})\\ =&D-d\end{aligned} \tag{B.39}$$

in an upper-bound on the delay of f through $\mathcal{C}_{\sigma_{f,n}}$ and 0 is obviously a lower-bound.

Then the delay bounds through the system marked as $\mathcal{S}''$ on Figure B.11 is

$$\begin{aligned}[d, D] + [0, D-d]\\ = [d, 2D-d]\end{aligned} \tag{B.40}$$

Last, the input of $\mathcal{S}''$ is packetized because the output a^* of the diamond ancestor is located after the packetizer within the input port. We can hence apply Theorem 2.5 and we obtain that $2D-d$ is also a delay bound through the concatenation of $\mathcal{S}''$ with the packetizer, *i.e.*, through $\mathcal{S}'$. □

B.2.9 Proof of Theorem 4.5

Proof of Theorem 4.5. Consider a system defined by Figure 4.20 and by Conditions (a) to (c) of Theorem 4.5. Take any $r > 0$, $b > 0$ and d_1, D_1, d_2, D_2 such that Conditions (d) to (e) of Theorem 4.5 are met. We first describe the adversarial model applied when $D_1 < d_2$.

Adversarial model for the case $D_1 < d_2$

We exhibit an adversarial model $\mathcal{M}_{D_1<d_2}$ for the sources and for the paths $\{P_j\}_j$ such that Properties 1/ to 5/ of Theorem 4.5 hold for $\mathcal{M}_{D_1<d_2}$.

Constants of $\mathcal{M}_{D_1<d_2}$ We define

$$J \triangleq d_2 - D_1 \tag{B.41}$$

And

$$D \triangleq d_2 \qquad d \triangleq D_1 \tag{B.42}$$

thus $d < D$. Note that $q \geq q_{\min}$ can be written

$$q \ \geq q_{\min} = \left\lfloor \frac{2rJ}{b} + 2 \right\rfloor + 1$$

With $J > 0$. Note that $q \ > \frac{2rJ}{b} + 2$ thus $(q-2)\frac{b}{r} > 2J$. Therefore, take any ϵ such that

$$\min\left(\frac{b}{r} - \frac{2}{q-2}J, J\right) > \epsilon > 0 \tag{B.43}$$

We further define

$$I \triangleq \max\left(\frac{q}{q-2}J, \frac{b}{r}\right) \tag{B.44}$$

$$\phi \triangleq I - J + \epsilon \tag{B.45}$$

and

$$\tau \triangleq q\phi \tag{B.46}$$

Finally, we consider a starting instant $x_1 > 0$ and for $i \in [\![1, q]\!]$, we define

$$x_i \triangleq (i-1)\phi + x_1 \tag{B.47}$$

Properties on the constants of $\mathcal{M}_{D_1<d_2}$ For $q > 3$, $\frac{q}{q-2} > 1$ thus by (B.44)

$$I > J > 0 \tag{B.48}$$

thus we also have $\phi > 0$ and $\tau > 0$ by (B.45) and (B.46). Furthermore,

$$\begin{aligned}
\tau - I &= q(I-J) + q\epsilon - I && \triangleright \text{ by } (B.45), (B.46)\\
&= (q-2)I - qJ + q\epsilon + I &&\\
&\geq qJ - qJ + q\epsilon + I && \triangleright \text{ by } (B.44)\\
&> I && \triangleright \text{ by } (\text{B}.43)
\end{aligned}$$

combined again with (B.44), this gives

$$\tau - I > \frac{b}{r} \tag{B.49}$$

For ϕ, we first have

$$\phi < I \tag{B.50}$$

because $\epsilon < J$ and

$$\phi < I + \frac{b}{r} - \frac{q}{q-2}J \tag{B.51}$$

because $\epsilon < \frac{b}{r} - \frac{2}{q-2}J$. By (B.44), I can take only one of two values. If $I = \frac{b}{r}$, then (B.50) gives $\phi < \frac{b}{r}$. If $I = \frac{q}{q-2}J$, then (B.51) gives $\phi < \frac{b}{r}$. We hence prove

$$\phi < \frac{b}{r} \tag{B.52}$$

Adversarial traffic generation at the source in $\mathcal{M}_{D_1<d_2}$ For each $i \in [\![1, q]\!]$, the source a in Figure 4.20 sends[1] a data unit $m^1_{i,k}$, of size b at the time instant $x_i + k\tau$ and $m^2_{j,k}$ of size b at the time instant $x_i + k\tau + I$ for all $k \in \mathbb{N}$.

Figure B.12 presents the traffic at the output of the source, focusing on two successive flows: f_i and f_{i+1} (with $i \leq q-1$). Their source profiles are periodic with a period τ and Figure B.12 focuses on the k-th period. For the flow f_i (solid-blue data units), the source generates the data unit $m^1_{i,k}$ at time $x_i + k\tau$, then sends $m^2_{i,k}$ after a duration I and it finally waits for the next period $(k+1)$ before it restarts the same profile and sends $m^1_{i,k+1}$. The source profile for flow f_{i+1} (dashed-red data units) is identical, but shifted by ϕ with respect

[1]If b is larger than the maximal packet length, then the source sends several data units simultaneously such that the sum of their length equal b. In this case, $m^1_{i,k}$ and $m^2_{i,k}$ represent the set of these data units.

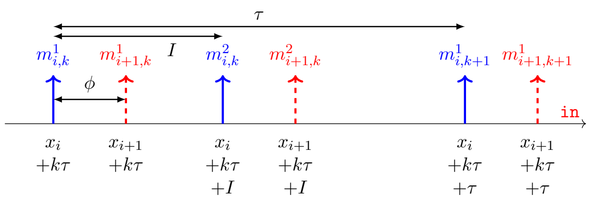


Figure B.12: Generation of data units for flows f_i and f_{i+1} $(i \leq q-1)$. Their traffic profile is periodic with period τ. The source sends a data unit for f_i at $x_i + k\tau$ for $k \in \mathbb{N}$, then it sends another data unit after a duration I and finally restarts at the next period. The profile for f_{i+1} is identical and shifted by ϕ with respect to the one of f_i $(x_{i+1} = x_i + \phi)$.

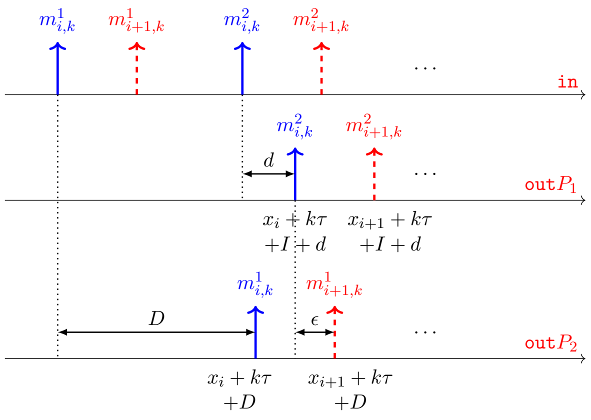


Figure B.13: Traffic profile for the flows f_i and f_{i+1} at the output of the two adversarial paths P_1 and P_2, in the model $\mathcal{M}_{D_1<d_2}$. P_1 drops all m^1 data units whereas P_2 drops all m^2 data units.

to the source profile for f_i, because $x_{i+1} = x_i + \phi$ by (B.47). By (B.45) and (B.43), $\phi < I$ thus $m^1_{i,k+1}$ is sent before $m^2_{i,k}$ as shown in the figure.

Properties on the traffic generation at the source in $\mathcal{M}_{D_1<d_2}$ By (B.44) and (B.49), the minimum time elapsed at the source between any two data units of f_i is larger than b/r, which shows that Property 1/ of Theorem 4.5 holds.

Adversarial paths in $\mathcal{M}_{D_1<d_2}$

- For any $k \in \mathbb{N}$ and any $i \in [\![1, q]\!]$, path P_1 drops the packet containing the data unit $m^1_{i,k}$ and forwards the packet containing the data unit $m^2_{i,k}$ with a delay d.
- For any $k \in \mathbb{N}$ and any $i \in [\![1, q]\!]$, path P_2 forwards the packet containing the data unit $m^1_{i,k}$ with a delay D and drops the packet containing the data unit $m^2_{i,k}$.
- Any other path P_j with $j \geq 3$ drops all packets.

Figure B.13 shows the trajectory at the output the two adversarial paths, focusing on period k and on flows f_i and f_{i+1}. Path $P1$ drops the packets containing the data units $m^1_{i,k}$ and $m^1_{i+1,k}$. It forwards those that contain $m^2_{i,k}$ and $m^2_{i+1,k}$ with a delay d. Similarly, P_2 drops $m^2_{i,k}$ and $m^2_{i+1,k}$ but forwards $m^1_{i,k}$ and $m^1_{i+1,k}$ with a delay D.

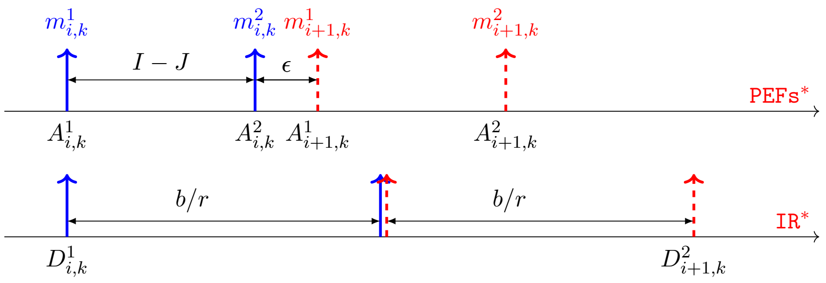


Figure B.14: Traffic profile for the flows f_i and f_{i+1} at the input of the IR (above) and at its output (below). To ease the lecture, the scale is not the same as in Figures B.12 and B.13.

Properties of the paths in $\mathcal{M}_{D_1<d_2}$ The delay of the non-lost packets through P_1 [resp., through P_2] equals $d = D_1$ [resp., $D = d_2$] that belongs to $[d_1, D_1]$ [resp., to $[d_2, D_2]$]. Thus the adversarial paths meet Properties 2/ and 4/ of Theorem 4.5.

Effect of the PEFs in $\mathcal{M}_{D_1<d_2}$ The set of parallel PEFs in Figure 4.20 receive the sum of the two paths outputs. As per its model in Section 4.2.2, each PEF forwards the first packet containing the data unit. For $i \in [\![1,q]\!]$, $k \in \mathbb{N}$ and $w \in \{1,2\}$, we denote by $A^w_{i,k}$ the time instant at which the unique packet containing the data unit $m^w_{i,k}$ exits the set of parallel PEFs.

By construction of the adversarial paths, for $i \in [\![1,q]\!]$ and $k \in \mathbb{N}$, only path P_1 forwards a packet containing the data unit $m^2_{i,k}$, released d after its emission by the source. Thus $m^2_{i,k}$ exits the PEFs as soon as the packet exits P_1. We obtain

$$\forall i \in [\![1,q]\!], \forall k \in \mathbb{N} \quad A^2_{i,k} = x_i + k\tau + I + d \tag{B.53}$$

Similarly with $m^1_{i,k}$ that is only forwarded by P_2,

$$\forall i \in [\![1,q]\!], \forall k \in \mathbb{N} \quad A^1_{i,k} = x_i + k\tau + D \tag{B.54}$$

The top line of Figure B.14 shows the trajectory at the output of the PEFs focusing on flows f_i and f_{i+1} and on the k-th period of the profile. Note that `PEFs`* is the output of the system denoted by S in Section 4.4.2 and is also the input of the IR (Figure 4.20).

Properties of the system S between `in` and `PEFs`* in $\mathcal{M}_{D_1<d_2}$ For $i \in [\![1,q]\!]$ and $n \in \mathbb{N}$ we note that

$$\begin{aligned} A^2_{i,k} - A^1_{i,k} &= I + d - D && \triangleright \text{ (B.53) and (B.54)} \\ &= I - J && \triangleright \text{ (B.41)} \\ &> 0 && \triangleright \text{ (B.48)} \end{aligned}$$

This proves that $m^2_{i,k}$ exit S after $m^1_{i,k}$ for any $i \in [\![1, q]\!]$ and any $k \in \mathbb{N}$. Also,

$$\begin{aligned}
A^1_{i,k+1} - A^2_{i,k} &= \tau + D - d - I && \triangleright \text{ (B.53) and (B.54)} \\
&= \tau - (I - J) \\
&= (q-1)(I - J) + q\epsilon && \triangleright \text{ (B.46) and (B.45)} \\
&> 0 && \triangleright\ q \geq 3
\end{aligned}$$

And this proves that for any $i \in [\![1, q]\!]$ and any $k \in \mathbb{N}$, $m^2_{i,k}$ exits S before $m^1_{i,k+1}$. Therefore, S is FIFO for f_i, for any $i \in [\![1, q]\!]$. Furthermore, each data unit is transported through exactly one path (either P_1 or P_2), thus S is also lossless. This proves that Property 5/ of Theorem 4.5 holds.

Last, we note that

$$\begin{aligned}
A^1_{i+1,k} - A^2_{i,k} &= x_{i+1} - x_i + D - d - I && \triangleright \text{ (B.53) and (B.54)} \\
&= \phi + J - I && \triangleright \text{ (B.41) and (B.47)} \\
&= \epsilon > 0 && \triangleright \text{ (B.45)}
\end{aligned} \tag{B.55}$$

Therefore, $m^1_{i+1,k}$, the first packet of the flow f_{i+1} in the k-th period exits the PEFs ϵ seconds after the second packet of the flow f_i in the k-th period, as described in Figure B.14.

Output of the IR in $\mathcal{M}_{D_1<d_2}$ For $i \in [\![1, q]\!]$, $n \in \mathbb{N}$ and $w \in \{1, 2\}$, we denote by $D^w_{i,k}$ the absolute time at which data unit $m^w_{i,k}$ leaves the IR.

The bottom line of Figure B.14 shows the release time of the data units out of the IR. Assume for example that the source has been idle for a while, then the regulator is empty and data unit $m^1_{i,k}$ can be released immediately without violating the shaping curve for f_i, thus $D^1_{i,k} = A^1_{i,k}$.

However, data unit $m^2_{i,k}$ arrives at the IR too soon with respect to the shaping curve σ_{f_i}. By applying the equations of the IR Le Boudec 2018, we note that the IR must delay $m^2_{i,k}$ and

$$\forall i \in [\![1, q]\!], \forall k \in \mathbb{N}, \quad D^2_{i,k} \geq D^1_{i,k} + b/r \tag{B.56}$$

By (B.55), data unit $m^1_{i+1,k}$ arrives after the data unit $m^2_{i,k}$. As the IR looks only at the head-of-line packet and is itself a FIFO system, we obtain

$$\forall i \in [\![1, q]\!], \forall k \in \mathbb{N}, \quad D^1_{i+1,k} \geq D^2_{i,k} \tag{B.57}$$

Combining Equations (B.56) and (B.57) gives, by induction,

$$\forall k \in \mathbb{N}, \quad D^2_{q,k} \geq D^1_{1,k} + q\frac{b}{r} \tag{B.58}$$

Now we note that

$$\begin{aligned}
A^1_{1,k+1} &= x_1 + (k+1)\tau + D && \triangleright \text{ (B.54)}\\
&= x_1 + k\tau + q\phi + D && \triangleright \text{ (B.46)}\\
&= x_q + k\tau + \phi + D && \triangleright \text{ (B.47)}\\
&= x_q + k\tau + I - J + \epsilon + D && \triangleright \text{ (B.45)}\\
&= x_q + k\tau + I + d + \epsilon && \triangleright \text{ (B.41)}\\
&= A^2_{q,k} + \epsilon && \triangleright \text{ (B.53)}
\end{aligned}$$

Therefore, the first data unit of the $(k+1)$-th period of f_1 arrives ϵ seconds after the second data unit of the k-th period of the last flow f_q. The IR being FIFO, we have

$$\forall k \in \mathbb{N}, \quad D^1_{1,k+1} \geq D^2_{q,k} \tag{B.59}$$

which, combined with (B.57), gives

$$\forall k \in \mathbb{N}, \quad D^1_{1,k+1} \geq D^1_{1,k} + q\frac{b}{r} \tag{B.60}$$

At period $k=0$, the network is empty and $D^1_{1,0} = A^1_{1,0} = x_1$. The induction of (B.60) thus gives

$$\forall k \in \mathbb{N}, \quad D^1_{1,k} \geq x_1 + kq\frac{b}{r} \tag{B.61}$$

And the delay, through the IR, suffered by the first data unit of the k-th period of the first flow f_1 is

$$\begin{aligned}
&D^1_{1,k} - A^1_{1,k}\\
&\geq x_1 + kq\frac{b}{r} - x_1 - k\tau - D && \triangleright \text{ (B.61) and (B.54)}\\
&\geq -D + kq\left(\frac{b}{r} - \phi\right) && \triangleright \text{ (B.46)}
\end{aligned}$$

By (B.52), $\frac{b}{r} - \phi > 0$. Thus the above delay lower-bound diverges as k increases and Property 3/ of the Theorem holds.

Adversarial model for the case $d_2 < D_1$

The adversarial model $\mathcal{M}_{d_2<D_1}$ follows the same principle as the adversarial model $\mathcal{M}_{D_1<d_2}$ described above. In the following, we detail only the differences.

Constants of $\mathcal{M}_{d_2<D_1}$ By assumption, $D_1 - d_2 > 0$. Furthermore, $q_{\min}$ now equals 3 and $q \geq q_{\min}$, $\left(\frac{q-2}{2}\right)\frac{b}{r} > 0$.

We hence select J such that

$$0 < J < \min\left(\frac{q-2}{2}\frac{b}{r}, D_1 - d_2\right) \tag{B.62}$$

And we re-define

$$D \triangleq D_1 \qquad d \triangleq D - J \tag{B.63}$$

As $J < D_1 - d_2 =$, $J > 0$, and $D_2 \geq D_1$ by on the indexes, we obtain $D_2 \geq D_1 > d > d_2$ thus

$$d \in [d_2, D_2] \tag{B.64}$$

By definition, $\frac{2}{q-2}J < \frac{b}{r}$, thus we define ϵ, I, ϕ, τ and x_i as in Appendix B.2.9, *i.e.*, per Equations (B.43), (B.44), (B.45), (B.46) and (B.47).

Properties on the constants of $\mathcal{M}_{d_2<D_1}$ None of the properties established in Appendix B.2.9 depends on the definition of J, d or D. They are all obtained thanks to the definitions of the other constants. As we re-use the same definitions, all the properties obtained in Appendix B.2.9 are also valid for $\mathcal{M}_{d_2<D_1}$.

Adversarial traffic generation at the source in $\mathcal{M}_{d_2<D_1}$ The adversarial model $\mathcal{M}_{d_2<D_1}$ uses the same traffic generation as $\mathcal{M}_{D_1<d_2}$. It is described in Appendix B.2.9 and summarized in Figure B.12.

Properties on the traffic generation at the source in $\mathcal{M}_{d_2<D_1}$ The traffic generation is not modified, thus the properties established in Appendix B.2.9 also hold for $\mathcal{M}_{d_2<D_1}$. In particular, Property 1/ of Theorem 4.5 holds.

Adversarial paths in $\mathcal{M}_{d_2<D_1}$ With respect to the model $\mathcal{M}_{D_1<d_2}$, the adversarial model $\mathcal{M}_{d_2<D_1}$ simply flips the the roles of each paths. Specifically,

- For any $k \in \mathbb{N}$ and any $i \in [\![1, q]\!]$, path P_1 forwards the packet containing the data unit $m^1_{i,k}$ with a delay D and drops the packet containing the data unit $m^2_{i,k}$.
- For any $k \in \mathbb{N}$ and any $i \in [\![1, q]\!]$, path P_2 drops the packet containing the data unit $m^1_{i,k}$ and forwards the packet containing the data unit $m^2_{i,k}$ with a delay d.
- Any other path P_j with $j \geq 3$ drops all packets.

The output of both paths is shown in Figure B.15. We can note the symmetry with Figure B.13.

Properties of the paths in $\mathcal{M}_{d_2<D_1}$ The packets not lost in P_1 have the same delay through P_1 equal to D. Similarly, the packets not lost in P_2 have the same delay through P_2 equal to d. Thus both P_1 and P_2 are FIFO and Property 4/ of Theorem 4.5 hold.

Furthermore, by (B.63), $D = D_1$ and by (B.64), $d \in [d_2, D_2]$. Thus Property 2/ holds.

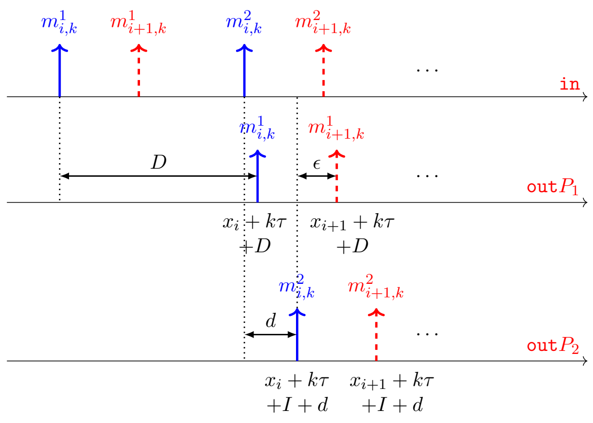


Figure B.15: Traffic profile for the flows f_i and f_{i+1} at the output of the two adversarial paths P_1 and P_2, in the model $\mathcal{M}_{d_2<D_1}$. P_1 drops all m^2 data units whereas P_2 drops all m^2 data units. With respect to Figure B.13, the roles of P_1 and P_2 have been exchanged.

Effect of the PEFs in $M_{d_2<D_1}$ As for the $\mathcal{M}_{D_1<d_2}$ model, each data unit arrives in a unique packet at the PEFs (m^1 data units arrive only through P_1 and m^2 data units arrive only through P^2). Thus the PEFs are transparent and forward the sum of both output traffic, `out`P_1 and `out`P_2 from Figure B.15. We observe that the sum of them gives the same output as on the first line of Figure B.14.

Therefore, all the remaining steps of the proof (properties on the system S, output of the IR with diverging delays) can be followed as in the model $\mathcal{M}_{D_1<d_2}$.

Adversarial model for the case $d_2 = D_1$

If $d_2 = D_1$, then $q_{\min} = 3$. By Condition (e) of Theorem 4.5, one of the two intervals $[d_1, D_1]$ or $[d_2, D_2]$ has a strictly positive length. Assume for example that $d_2 < D_2$. Then we simply select d_2' such that

$$d_2 < d_2' < \min\left(D_2, D_1 + \frac{b}{2r}\right) \tag{B.65}$$

We obtain

$$\left\lfloor \frac{2r\,|d_2' - D_1|^+}{b} + 2 \right\rfloor + 1 = 3$$

because

$$\frac{2r\,|d_2' - D_1|^+}{b} < 1$$

by choice of d_2'. This means that we can apply model $\mathcal{M}_{d_2'<D_1}$ with parameters $q, r, b, d_1, d_2', D_1, D_2$ and the same number of flows ($q \geq 3$). This model will provide Properties 1/ to 5/ of Theorem 4.5 for the choice of parameters $q, r, b, d_1, d_2', D_1, D_2$, thus providing Properties 1/ to 5/ of Theorem 4.5 for the choice parameters r, b, d_1, d_2, D_1, D_2.

□

B.2.10 Proof of Corollary 4.4

Proof of Corollary 4.4. We simply construct $\mathcal{S}$ as a system that contains a packet-replication function (PRF), two alternative paths P_1, P_2 and a set of PEFs, as in Figure 4.20. We then apply Theorem 4.5 with lossless and FIFO paths P_1 and P_2 that have both the same delay interval $[d_1, D_1] = [d_2, D_2] = [0, D_{\max}]$. □

B.2.11 Proof of Theorem 4.6

Proof of Theorem 4.6. We denote by $[d^\dagger, D^\dagger]$ the lower and upper delay bounds of the non-lost data units through the system $\mathcal{S}^\dagger$ between the output of a and the output of the POF (Figure 4.23).

The output of a vertex corresponds to the output of the packetizer, thus the flow aggregate $\mathcal{F}$ is packetized at the output of vertex a. We can hence define the packet sequence (A, L, F) for the aggregate $\mathcal{F}$ as in Le Boudec 2018, §II.A:

- A is the sequence of the arrival times at the observation point a^* for the data units that belong to the flow aggregate $\mathcal{F}$. A is a wide-sense increasing sequence. *I.e.,* A_n is the arrival time at a^* of the n-th data unit of the aggregate $\mathcal{F}$.
- L is the sequence of packet length for the above data units. *I.e.,* L_n is the length of the packet that transports the n-th data unit that arrives at a^* and belongs to $\mathcal{F}$.
- F is the sequence of flow identifiers for the above data units. *I.e.,* $F_n = f$ means that the n-th data unit of the aggregate $\mathcal{F}$ at a^* belongs to flow f.

We also define the Π^f regulator for each flow f of $\mathcal{F}$ that corresponds to the shaping curve $\sigma_{f,n}$ of the IR $\texttt{REG}_n(\mathcal{F}, a)$ Le Boudec 2018, IV.A. By configuration of $\texttt{REG}_n(\mathcal{F}, a)$, each flow f of the aggregate is $\sigma_{f,n}$-constrained thus Π^f-regular at a^*, the input of the systems $\mathcal{S}$, $\mathcal{S}^\dagger$ and S'.

– If $\mathcal{S}$ is **lossless** for $\mathcal{F}$, we apply Mohammadpour, Le Boudec 2021, Theorem 4 and obtain $d^\dagger = d$ and $D^\dagger = D$.

Then, by definition of the POF and considering its configuration $\texttt{POF}_n(\{f\}, a)$, system $\mathcal{S}^\dagger$ is FIFO and lossless for the aggregate $\mathcal{F}$ processed by the regulator. Therefore, applying Le Boudec 2018, Theorem 5 gives $d' = d^\dagger$ and $D' = D^\dagger$.

– If $\mathcal{S}$ is **not lossless** for $\mathcal{F}$, then the application of Mohammadpour, Le Boudec 2021, Theorem 4 gives $d^\dagger = d$ and $D^\dagger = D + T$.

Then, by definition of the POF and considering its configuration $\texttt{POF}_n(\{f\}, a)$, system $\mathcal{S}^\dagger$ is FIFO but not lossless for the aggregate $\mathcal{F}$ processed by the regulator.

Like in the proof of Lemma B.2, we decompose the packet sequence (A, L, F) at the input of $\mathcal{S}^\dagger$ into the sub-sequences (A_1, L_1, F_1) and (A_2, L_2, F_2) that correspond respectively to the data units that are not lost inside $\mathcal{S}^\dagger$ and to the data units that are lost inside $\mathcal{S}^\dagger$.

We consider the cumulative functions $R_{f,1}$ and $R_{f,2}$ of each flow f that correspond to the sequence (A_1, L_1, F_1) and (A_2, L_2, F_2), respectively. Then, $R_f \triangleq R_{f,1} + R_{f,2}$, the overall cumulative function, corresponds to the sequence (A, L, F) for flow f thus R_f is $\sigma_{f,n}$-constrained. Re-using an argument from the proof of Lemma B.2, the cumulative sub-function $R_{f,1}$ remains $\sigma_{f,n}$-constrained. Thus flow f in sub-sequence (A_1, L_1, F_1) remains Π^f-regular.

Therefore, we apply Le Boudec 2018, Theorem 5 on the sub-sequence (A_1, L_1, F_1) and we obtain that for the corresponding IR output sequence (D_1, L_1, F_1), the delay through $\mathcal{S}'$ verifies $d' = d^\dagger$ and $D' = D^\dagger$. The delay of the non-lost data units through $\mathcal{S}'$ is hence within $[d', D'] = [d, D + T]$. □

B.3 Proofs of Chapter 5

B.3.1 Proof of Proposition 5.1

Proof of Proposition 5.1. Denote by $d \triangleq d_{g\leftarrow i}$. We start with a lemma

Lemma B.13 (Changing the clock for the measure of a duration, general result) *If $\chi^{\mathcal{H}_i}$ is the measure of a duration with clock $\mathcal{H}_i$, then the measure $\chi^{\mathcal{H}_g}$ of the same duration with $\mathcal{H}_g$ is bounded by:*

$$(d\overline{\oslash}d)(\chi^{\mathcal{H}_i}) \quad \leq \quad \chi^{\mathcal{H}_g} \quad \leq (d \oslash d)(\chi^{\mathcal{H}_i}) \tag{B.66}$$

where $\oslash$ is the min-plus deconvolution[2] *(Definition 2.3), $\overline{\oslash}$ is the max-plus deconvolution*[3] *and d is the relative time function $d \triangleq d_{g\leftarrow i}$.*

Proof of Lemma B.13. Consider the event ω_s [resp., ω_e] that corresponds to the start [resp., the end] of the considered duration. Denote by $t_s^{\mathcal{H}_i}$ [resp., $t_e^{\mathcal{H}_i}$] the measure with $\mathcal{H}_i$ of the time instant (date) of event w_s [resp., of event w_e]. Then

$$\chi^{\mathcal{H}_i} = t_e^{\mathcal{H}_i} - t_s^{\mathcal{H}_i} \tag{B.67}$$

Similarly, denote by $t_s^{\mathcal{H}_g}$ [resp., $t_e^{\mathcal{H}_g}$] the measure with $\mathcal{H}_g$ of the time instant (date) of ω_s [resp., of ω_e] with $\mathcal{H}_g$. As $t_s^{\mathcal{H}_i}$ and $t_s^{\mathcal{H}_g}$ are the measure of the time instant of the same event, but with two different clocks, we have

$$t_s^{\mathcal{H}_g} = d_{g\leftarrow i}(t_s^{\mathcal{H}_i}) \quad \text{and} \quad t_e^{\mathcal{H}_g} = d_{g\leftarrow i}(t_e^{\mathcal{H}_i}) \tag{B.68}$$

Then

$$\chi^{\mathcal{H}_g} = t_e^{\mathcal{H}_g} - t_s^{\mathcal{H}_g} \tag{B.69}$$

$$= d(t_e^{\mathcal{H}_i}) - d(t_s^{\mathcal{H}_i}) \qquad = d(t_s^{\mathcal{H}_i} + \chi^{\mathcal{H}_i}) - d(t_s^{\mathcal{H}_i}) \tag{B.70}$$

One one hand, $d(t_s^{\mathcal{H}_i} + \chi^{\mathcal{H}_i}) - d(t_s^{\mathcal{H}_i}) \leq \sup_{u\geq 0} d(u + \chi^{\mathcal{H}_i}) - d(u)$ because $t_s^{\mathcal{H}_i} \geq 0$. On the other hand, $d(t_s^{\mathcal{H}_i} + \chi^{\mathcal{H}_i}) - d(t_s^{\mathcal{H}_i}) \geq \inf_{u\geq 0} d(u + \chi^{\mathcal{H}_i}) - d(u)$ because again $t_s^{\mathcal{H}_i} \geq 0$. We hence have

$$\inf_{u\geq 0} d(u + \chi^{\mathcal{H}_i}) - d(u) \leq \chi^{\mathcal{H}_g} \qquad \leq \sup_{u\geq 0} d(u + \chi^{\mathcal{H}_i}) - d(u) \tag{B.71}$$

$$(d\overline{\oslash}d)(\chi^{\mathcal{H}_i}) \quad \leq \quad \chi^{\mathcal{H}_g} \qquad \leq (d \oslash d)(\chi^{\mathcal{H}_i}) \tag{B.72}$$

[2] $\mathfrak{f} \oslash \mathfrak{g} : t \mapsto \sup_{u\geq 0} \mathfrak{f}(t + u) - \mathfrak{g}(u)$
[3] $\mathfrak{f}\overline{\oslash}\mathfrak{g} : t \mapsto \inf_{u\geq 0} \mathfrak{f}(t + u) - \mathfrak{g}(u)$

□

By Equation 5.6, for any $t \geq 0, u \geq 0$, $d(t+u)-d(t) \leq \rho u+\eta$ and $d(t+u)-d(t) \geq (u-\eta)/\rho$. If, the network is also synchronized, then $d(t+u) - d(t) \leq |d(t+u) - t| + |t - d(t)| \leq 2\Delta$, and $d(t+u) - d(t) \geq -|d(t+u) - t| - |t - d(t)| \geq -2\Delta$. Last, as $d_{g\to i}$ is strictly increasing, then $\chi^{\mathcal{H}_g} \geq 0$ because $\chi^{\mathcal{H}_i} \geq 0$. □

B.3.2 Proof of Proposition 5.2

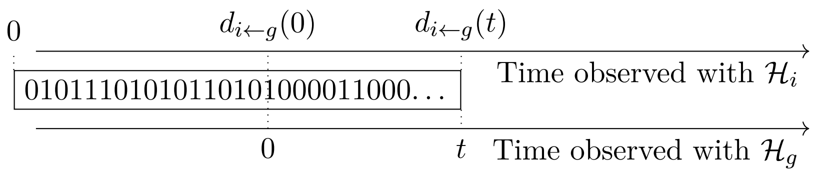


Figure B.16: Case where the initial offset of the local clock $\mathcal{H}_i$ is positive compared to the reference clock $\mathcal{H}_g$.

Proof of Proposition 5.2. Note that the same amount of data of f enter S between the time instants measured as t_1 and t_2 using $\mathcal{H}_g$ and between the time instants measured as $d_{i\leftarrow g}(t_1)$ and $d_{i\leftarrow g}(t_2)$ using $\mathcal{H}_i$. We now split the proof into two situations.

Case $d(0) \geq 0$: If the origin of $\mathcal{H}_i$ is before the origin of $\mathcal{H}_g$, then the situation is as in Figure B.16, where we represent face-to-face the two clocks. For any $t \geq 0$, the number of bits observed using $\mathcal{H}_i$ between 0 and $d(t)$ equals the number of bits observed between 0 and $d(0)$, plus the number of bits observed between $d(0)$ and $d(t)$. That second term also equals the number of bits observed between 0 and t using $\mathcal{H}_g$ (Figure B.16), i.e., $R^{\mathcal{H}_g}(t)$. Hence,

$$\forall t \geq 0, \qquad R^{\mathcal{H}_i}(d(t)) = R^{\mathcal{H}_i}(d(0)) + R^{\mathcal{H}_g}(t) \tag{B.73}$$

Due to the definition of T_{start}, no bit could have been sent before the time measured as 0 using $\mathcal{H}_g$. Consequently, $R^{\mathcal{H}_i}(d(0)) = 0$, and we have the result.

Case $d(0) < 0$: If the origin of clock $\mathcal{H}_i$ is after the origin of clock $\mathcal{H}_g$, then by symmetry, we simply flip the pair and obtain the result from case *a)*. □

B.3.3 Proof of Proposition 5.3

Proof of Proposition 5.3. Take $t \geq s \geq 0$ and define $\tau = t - s$, then

$$\begin{aligned}
&R^{\mathcal{H}_g}(s+\tau) - R^{\mathcal{H}_g}(s) \\
&= R^{\mathcal{H}_i}(d(s+\tau)) - R^{\mathcal{H}_i}(d(s)) && \triangleright \text{ Proposition 5.2} \\
&\leq \alpha^{\mathcal{H}_i}(d(s+\tau) - d(s)) && \triangleright\ \alpha^{\mathcal{H}_i} \text{ is an arrival curve when observed with } \mathcal{H}_i \\
&\leq \sup_{s'\geq 0}\left[\alpha^{\mathcal{H}_i}(d(s'+\tau) - d(s'))\right] && \triangleright\ s \in \{s' | s' \geq 0\} \\
&\leq \alpha^{\mathcal{H}_i}(\sup_{s'\geq 0}[d(s'+\tau) - d(s')]) && \triangleright\ \alpha^{\mathcal{H}_i} \text{ is wide-sense increasing} \\
&\leq \alpha^{\mathcal{H}_i}((d \oslash d)(\tau))
\end{aligned}$$

Also, per Equation (5.5), $(d \oslash d)(\tau) \leq \rho\tau + \eta$, hence $(d \oslash d)(\tau)$ is always finite in our model. □

B.3.4 Proof of Proposition 5.4

Proof of Proposition 5.4. Take $t \geq 0$. Then

$$\begin{aligned}
R^{\mathcal{H}_g}_{f,w^{\text{out}}}(t) &= R^{\mathcal{H}_i}_{f,w^{\text{out}}}(d(t)) && \triangleright \text{ Proposition 5.2} \\
&\geq \inf_{0\leq\sigma\leq d(t)}\left[R^{\mathcal{H}_i}_{f,w^{\text{in}}}(\sigma) + \beta^{\mathcal{H}_i}(d(t)-\sigma)\right] && \triangleright\ \beta^{\mathcal{H}_i} \text{ is a service curve observed with } \mathcal{H}_i
\end{aligned}$$

The function d is a permutation of $\mathbb{R}$, thus for any time instant $\sigma \geq d(0)$ measured with $\mathcal{H}_i$, we note $\sigma = d(s)$ with s the measure using $\mathcal{H}_g$. Hence,

$$\begin{aligned}
&R^{\mathcal{H}_g}_{f,w^{\text{out}}}(t) \\
&\geq \inf_{d^{-1}(0)\leq s\leq t}\left[R^{\mathcal{H}_i}_{f,w^{\text{in}}}(d(s)) + \beta^{\mathcal{H}_i}(d(t) - d(s))\right]
\end{aligned}$$

If $t < T_{\text{start}}$ (defined in Section 5.2.1): the result holds because no bit has been transmitted: $R^{\mathcal{H}_g}_{f,w^{\text{out}}}(t) = 0$; $\forall s \leq t < T_{\text{start}}, R^{\mathcal{H}_i}_{f,w^{\text{in}}}(d(s)) = 0$ and $\inf_{d^{-1}(0)\leq s\leq t}\beta^{\mathcal{H}_i}(d(t)-d(s)) = 0$, obtained for $s = t$.

If $t \geq T_{\text{start}}$: Call $A : s \mapsto R^{\mathcal{H}_i}_{f,w^{\text{in}}}(d(s)) + \beta^{\mathcal{H}_i}(d(t) - d(s))$. As $T_{\text{start}} \geq d^{-1}(0)$, we can split the domain of the $\inf_s A(s)$ into the two cases $s \leq T_{\text{start}}$ and $s \geq T_{\text{start}}$, the result being the minimum of the two obtained inf.

For $s < T_{\text{start}}$, $\beta^{\mathcal{H}_i}(d(t) - d(s)) \geq \beta^{\mathcal{H}_i}(d(t) - d(T_{\text{start}}))$ because d and $\beta^{\mathcal{H}_i}$ are both increasing functions. On the other hand, based on the assumption on T_{start}, $R^{\mathcal{H}_i}(d(s)) = R^{\mathcal{H}_i}(d(T_{\text{start}}))$ as both quantities equal 0 bit.

Consequently,

$$\inf_{d^{-1}(0)\leq s\leq T_{\text{start}}} A(s) \geq A(T_{\text{start}})$$

Hence $\inf_s A(s)$ is obtained for $s \in [T_{\text{start}}, t]$, i.e.,

$$R^{\mathcal{H}_g}_{f,w^{\text{out}}}(t) \geq \inf_{T_{\text{start}} \leq s \leq t} A(s)$$

By definition $T_{\text{start}} \geq 0$, so $[T_{\text{start}}, t] \subset [0, t]$, hence

$$\begin{aligned}
R^{\mathcal{H}_g}_{f,w^{\text{out}}}(t) &\geq \inf_{0 \leq s \leq t} A(s) \\
&\geq \inf_{0 \leq s \leq t} \left[R^{\mathcal{H}_i}_{f,w^{\text{in}}}(d(s)) + \beta^{\mathcal{H}_i}(d(t) - d(s)) \right] \\
&\geq \inf_{0 \leq s \leq t} \left[R^{\mathcal{H}_i}_{f,w^{\text{in}}}(d(s)) + \inf_{u|0 \leq u} \beta^{\mathcal{H}_i}(d(t-s+u) - d(u)) \right] && \triangleright\ s \in \{u|0 \leq u\} \\
&\geq \inf_{0 \leq s \leq t} \left[R^{\mathcal{H}_i}_{f,w^{\text{in}}}(d(s)) + \beta^{\mathcal{H}_i} \left[\inf_{u|0 \leq u} (d(t-s+u) - d(u)) \right] \right] && \triangleright\ \beta^{\mathcal{H}_i} \text{wide-sense increasing} \\
&\geq \inf_{0 \leq s \leq t} \left[R^{\mathcal{H}_i}(d(s)) + \beta^{\mathcal{H}_i}((d \overline{\oslash} d)(t-s)) \right]
\end{aligned}$$

With $\overline{\oslash}$ the max-plus deconvolution [Le Boudec, Thiran 2001, Def 3.2.2]. By Proposition 5.2, $R^{\mathcal{H}_i}_{f,w^{\text{in}}}(d(s)) = R^{\mathcal{H}_g}_{f,w^{\text{in}}}(s)$, hence

$$R^{\mathcal{H}_g}_{f,w^{\text{out}}}(t) \geq \inf_{0 \leq s \leq t} \left[R^{\mathcal{H}_g}(s) + \beta^{\mathcal{H}_i}((d \overline{\oslash} d)(t-s)) \right]$$

which proves that the system S offers the function $t \mapsto \beta^{\mathcal{H}_i}((d \overline{\oslash} d)(t-s))$ as a service curve when observed with $\mathcal{H}_g$. □

B.3.5 Proof of Proposition 5.5

Proof of Proposition 5.5. Consider a flow f that crosses a system S and a regulator, in sequence. Assume that the source of the flow is at the input to system S and call $\mathcal{H}_\phi$ the clock used to define the arrival curve $\alpha^{\mathcal{H}_\phi} = \gamma_{r,b}$ at the source. Assume the regulator is a PFR and let $\mathcal{H}_{\text{Reg}} \neq \mathcal{H}_\phi$ be its clock. Since the PFR is non-adapted, it is configured with shaping curve $\sigma^{\mathcal{H}_{\text{Reg}}} = \alpha^{\mathcal{H}_\phi}$.

Let the flow be greedy after T_{start} and generate packets of size ℓ bits, with $\ell \leq b$. Its cumulative arrival function at the source, measured in $\mathcal{H}_\phi$, is

$$R^{\mathcal{H}_\phi}_\phi(t) = \left\lfloor \frac{\alpha^{\mathcal{H}_\phi}(|t - T_{\text{start}}|^+)}{\ell} \right\rfloor \ell \tag{B.74}$$

where $\lfloor \cdot \rfloor$ is the floor function. The fact that this flow satisfies the arrival-curve constraint $\alpha^{\mathcal{H}_\phi}$ in $\mathcal{H}_\phi$ follows from [Le Boudec 2001, Thm III.2]. Also, system S provides a delay bound $D^{\mathcal{H}_{\text{TAI}}}_{f,S_k}$ in $\mathcal{H}_{\text{TAI}}$ thus, by Section 5.3.1, a bound $D^{\mathcal{H}_\phi}_{f,S_k} = \rho D^{\mathcal{H}_{\text{TAI}}}_{f,S_k} + \eta$ in $\mathcal{H}_\phi$. Let $R^{\mathcal{H}_\phi}_{\text{PFR}^{\text{in}}}$ be the cumulative arrival function of flow f at the input of the PFR, observed with $\mathcal{H}_\phi$. Thus, for every t,

$$R^{\mathcal{H}_\phi}_\phi(t - D^{\mathcal{H}_\phi}) \leq R^{\mathcal{H}_\phi}_{\text{PFR}^{\text{in}}}(t) \tag{B.75}$$

Let $d_{\text{Reg}\leftarrow\phi}(t) = t/\rho$ be the relative time function between $\mathcal{H}_\phi$ and $\mathcal{H}_{\text{Reg}}$. The function meets the conditions of Equation (5.5). Using Proposition 5.2, the cumulative arrival function of flow f at the input of the PFR, when observed with the PFR clock $\mathcal{H}_{\text{Reg}}$, is given by

$$R^{\mathcal{H}_{\text{Reg}}}_{\text{PFR}^{\text{in}}}(\tau) = R^{\mathcal{H}_{\phi}}_{\text{PFR}^{\text{in}}}\left(d^{-1}_{\text{Reg}\leftarrow\phi}(\tau)\right) = R^{\mathcal{H}_{\phi}}_{\text{PFR}^{\text{in}}}(\rho\tau) \tag{B.76}$$

Network calculus results are valid as long as all the notions are in the same time reference. In particular, the PFR can be modeled as a fluid greedy shaper followed by a packetizer, and the latter can be ignored for delay computations. The output of the fluid greedy shaper, observed with $\mathcal{H}_{\text{Reg}}$, is thus

$$R^{\mathcal{H}_{\text{Reg}}}_{\text{PFR}^{*}} = R^{\mathcal{H}_{\text{Reg}}}_{\text{PFR}^{\text{in}}} \otimes \gamma_{r,b}$$

It follows that, for all τ,

$$R^{\mathcal{H}_{\text{Reg}}}_{\text{PFR}^{*}}(\tau) \leq R_{\text{PFR}^{\text{in}}}\mathcal{H}_{\text{Reg}}(T_{\text{start}}) + \gamma_{r,b}(\tau - T_{\text{start}}) = r|\tau - T_{\text{start}}|^{+} + b \tag{B.77}$$

by definition of T_{start}.

We now show

Lemma B.14
For any $e > 0$, the worst-case delay through the PFR, measured in $\mathcal{H}_{\text{Reg}}$, is $\geq e$.

Proof of Lemma B.14. From (B.77), it follows that, for $\tau \geq T_{\text{start}}$,

$$R^{\mathcal{H}_{\text{Reg}}}_{\text{PFR}^{*}}(\tau + e) \leq r(\tau + e - T_{\text{start}}) + b \tag{B.78}$$

Combine Eqs (B.74)–(B.78) and obtain

$$\begin{aligned} R^{\mathcal{H}_{\text{Reg}}}_{\text{PFR}^{*}}(\tau + e) - R^{\mathcal{H}_{\text{Reg}}}_{\text{PFR}^{\text{in}}}(\tau) &\leq r(\tau + e - T_{\text{start}}) + b - R^{\mathcal{H}_{\phi}}(\rho\tau - D^{\mathcal{H}_{\phi}}_{f,S_k}) \\ &= r(\tau + e - T_{\text{start}}) + b - \left\lfloor \frac{r(\rho\tau - D^{\mathcal{H}_{\phi}}_{f,S_k} T_{\text{start}}) + b}{\ell} \right\rfloor \ell \\ &\leq r(\tau + e - T_{\text{start}}) + b - (r(\rho\tau - D^{\mathcal{H}_{\phi}}_{f,S_k} - T_{\text{start}}) + b) + \ell \\ &= r(1-\rho)\tau + re + rD^{\mathcal{H}_{\phi}}_{f,S_k} + \ell \end{aligned} \tag{B.79}$$

Thus $R^{\mathcal{H}_{\text{Reg}}}_{\text{PFR}^{*}}(\tau + e) - R^{\mathcal{H}_{\text{Reg}}}_{\text{PFR}^{\text{in}}}(\tau) < 0$ whenever $\tau > \left(\frac{re + rD^{\mathcal{H}_{\phi}}_{f,S_k} + \ell}{(\rho-1)r} \vee T_{\text{start}}\right)$; it follows that the delay, measured with $\mathcal{H}_{\text{Reg}}$, for packets arrived at the PFR after time $\left(\frac{re + rD^{\mathcal{H}_{\phi}}_{f,S_k} + \ell}{(\rho-1)r} \vee T_{\text{start}}\right)$ is larger than e. □

Lemma B.14 holds for any arbitrary $e > 0$, therefore the delay measured with $\mathcal{H}_{\text{Reg}}$ is unbounded. By Section 5.3.1, this also proves that the delay is not bounded when viewed from any clock of the network.

For the IR, the same adversarial example applies because an IR processing only one flow has the same behavior as a PFR [Le Boudec 2018].

□

B.3.6 Proof of Proposition 5.6

Proof of Proposition 5.6. Consider a system S and assume that S is the k-th hop for flow f. We note $S = S_k$ as in Figure 5.9. When observed with $\mathcal{H}_{\mathtt{REG}_{k-1}}$, the flow has the arrival curve $\alpha^{\mathcal{H}_{\mathtt{REG}_{k-1}}}_{(k-1)^*} = \sigma_{k-1}$ at the output of $\mathtt{REG}_{k-1}$. We now apply Table 5.1 with $\mathcal{H}_g = \mathcal{H}_{\mathtt{REG}_k}$ and $\mathcal{H}_i = \mathcal{H}_{\mathtt{REG}_{k-1}}$. We obtain that, when observed with the clock of the next regulator, $\mathcal{H}_{\mathtt{REG}_k}$, the flow leaves $\mathtt{REG}_{k-1}$ with a leaky-bucket arrival curve $\alpha^{\mathcal{H}_{\mathtt{REG}_k}}_{(k-1)^*}$ of rate $\rho r_{\mathtt{REG}_{k-1}}$ and burst $\mathtt{REG}_{k-1} + \eta r_{\mathtt{REG}_{k-1}}$.

From the configuration of $\mathtt{REG}_{k-1}$ and $\mathtt{REG}_k$, we note that $\alpha^{\mathcal{H}_{\mathtt{REG}_k}}_{k} = \alpha^{\mathcal{H}_{\mathtt{REG}_k}}_{(k-1)^*} \leq \sigma_k$. Consequently, all the conditions for the shaping-for-free property are met when observed with clock $\mathcal{H}_{\mathtt{REG}_k}$. We apply the respective theorems for both the PFR [Le Boudec, Thiran 2001, Thm 1.5.2] and the IR [Le Boudec 2018, Thm 5] using this clock. An upper bound on the delay for the flow through the system S$_k$ as measured with $\mathcal{H}_{\mathtt{REG}_k}$ is $\rho D^{\mathcal{H}_{\text{TAI}}}_{f,S_k} + \eta$ (Proposition 5.1). Applying the shaping-for-free property, this is also a valid delay bound for the flow trough the whole hop (S_k followed by regulator), when measuring with $\mathcal{H}_{\mathtt{REG}_k}$. To obtain a delay bound back in the measurement clock $\mathcal{H}_{\text{TAI}}$, we apply again Proposition 5.1, which gives the result. □

B.3.7 Proof of Proposition 5.7

We first establish the following lemma.

Lemma B.15
Assume that $\alpha_{2,(k-1)^*} = \gamma_{\rho r_0, b_{2,(k-1)^*}}$ is an arrival curve for the flow at the input of the S_k, observed in $\mathcal{H}_{TAI}$. Then

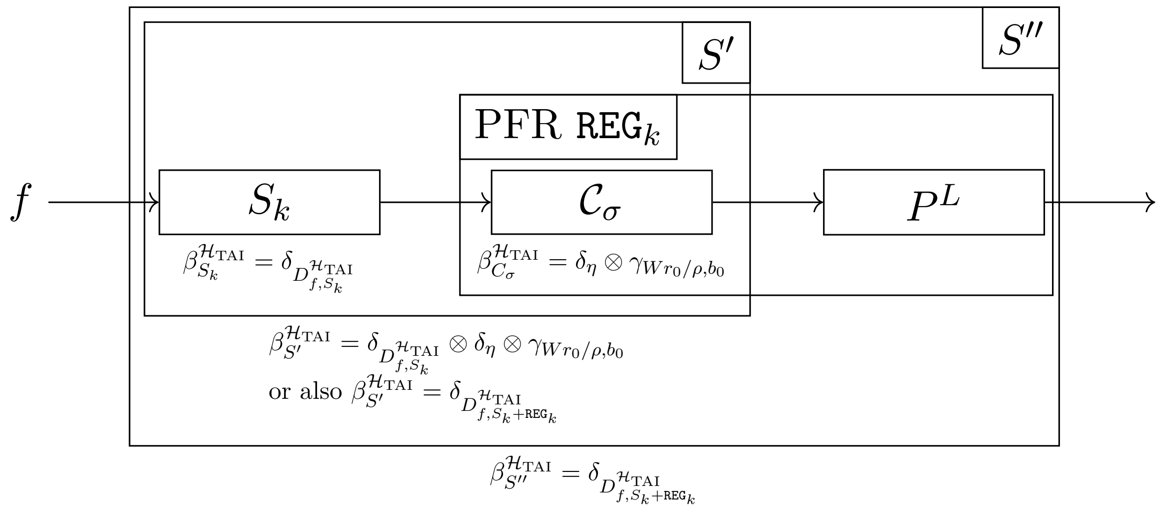
1. $D^{\mathcal{H}_{TAI}}_{f,S_k+REG_k} = D^{\mathcal{H}_{TAI}}_{f,S_k} + \eta(1+\rho) + \frac{b_{2,(k-1)^*} - b_0 - \eta W r_0}{\rho r_0}\frac{\rho^2-1}{W-1}$ *is a TAI delay bound for the flow through the concatenation of S_k and REG_k.*

2. $\gamma_{\rho r_0, b_{2,(k-1)^*}+\rho r_0 \cdot D^{\mathcal{H}_{TAI}}_{f,S_k+REG_k}}$ *is an arrival curve for the flow observed in $\mathcal{H}_{TAI}$ at the output of REG_k.*

Proof of Lemma B.15. **(1)** The shaping curve of the PFR $\mathtt{REG}_k$ is $\sigma_k = \gamma_{Wr_0,b_0}$. σ_k is concave and $\sigma(0^+) = b_0 \geq l_{\max}$. And the input of the PFR is packetized by assumption in Section 5.5.1. Therefore, the PFR $\mathtt{REG}_k$ can be modeled as a (fluid) greedy shaper $\mathcal{C}_\sigma$ followed by a packetizer P^L as in Figure B.17 [Le Boudec, Thiran 2001, Thm. 1.7.3]. Denote by S' [resp., S''] the concatenation of S_k and $\mathcal{C}_\sigma$ [resp., of $S_k, \mathcal{C}_\sigma$ and P^L] as in Figure B.17.

The greedy shaper $\mathcal{C}_\sigma$ provides to f the service curve $\beta^{\mathcal{H}_{\mathtt{REG}_k}}_{\mathcal{C}_\sigma} = \sigma = \gamma_{Wr_0,b_0}$ when observed with its internal clock, $\mathcal{H}_{\mathcal{C}_\sigma} = \mathcal{H}_{\mathtt{REG}_k}$ [Le Boudec, Thiran 2001, Cor. 1.5.1]. By applying Proposition 5.4 for the non-synchronized time model, we obtain that $\mathcal{C}_\sigma$ offers to f the service curve $\beta^{\mathcal{H}_{\text{TAI}}}_{\mathcal{C}_\sigma} = \delta_\eta \otimes \gamma_{Wr_0/\rho,b_0}$ when observed with $\mathcal{H}_{\text{TAI}}$ (Table 5.2). This service curve is shown under the $\mathcal{C}_\sigma$ box in Figure B.17.

$D^{\mathcal{H}_{\text{TAI}}}_{f,S_k}$ is a delay bound for f through S_k, observed with $\mathcal{H}_{\text{TAI}}$ and obtained through Step 2 of the ADAM method. S_k is causal, lossless and FIFO for f. Thus its provides to f the



S'
S''
PFR $\mathtt{REG}_k$
f
S_k
$\mathcal{C}_\sigma$
P^L
$\beta_{S_k}^{\mathcal{H}_{\mathrm{TAI}}} = \delta_{D_{f,S_k}^{\mathcal{H}_{\mathrm{TAI}}}}$
$\beta_{\mathcal{C}_\sigma}^{\mathcal{H}_{\mathrm{TAI}}} = \delta_\eta \otimes \gamma_{Wr_0/\rho, b_0}$
$\beta_{S'}^{\mathcal{H}_{\mathrm{TAI}}} = \delta_{D_{f,S_k}^{\mathcal{H}_{\mathrm{TAI}}}} \otimes \delta_\eta \otimes \gamma_{Wr_0/\rho, b_0}$
or also $\beta_{S'}^{\mathcal{H}_{\mathrm{TAI}}} = \delta_{D_{f,S_k+\mathtt{REG}_k}^{\mathcal{H}_{\mathrm{TAI}}}}$
$\beta_{S''}^{\mathcal{H}_{\mathrm{TAI}}} = \delta_{D_{f,S_k+\mathtt{REG}_k}^{\mathcal{H}_{\mathrm{TAI}}}}$

Figure B.17: Network-calculus model of the system S_k followed by the PFR $\mathtt{REG}_k$: the PFR can be modeled as a fluid greedy-shaper $\mathcal{C}_\sigma$ followed by a packetizer P^L. Below each box we show one (or two) service curves that the element provides to f when observed in $\mathcal{H}_{\mathsf{TAI}}$.

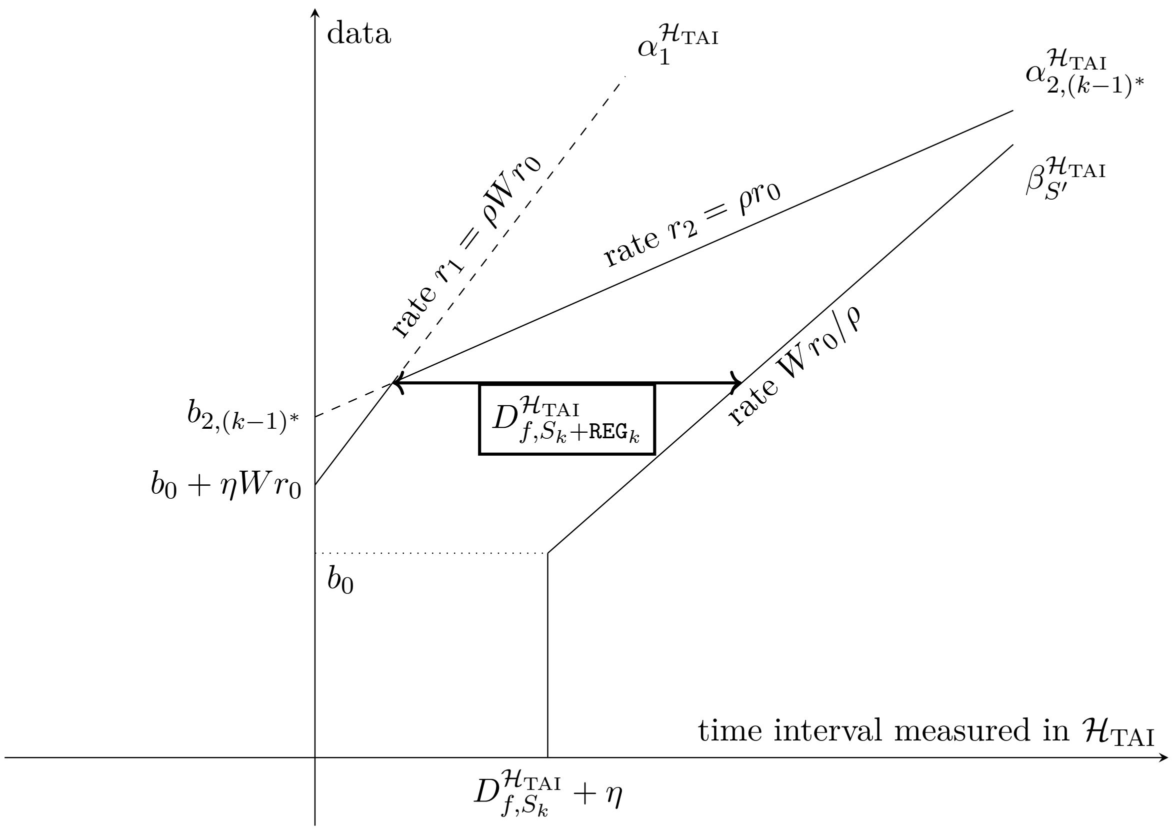


Figure B.18: TAI delay bound computation for the flow through the system S' of Figure B.17. The knowledge of $\alpha_{2,k}$ is required to provide a bounded delay whereas the knowledge of α_1 helps having a tighter delay bound.

service cure the service-curve $\delta_{D^{\mathcal{H}_{\mathrm{TAI}}}_{f,S_k}}$ Le Boudec, Thiran 2001, Prop. 1.3.3 when observed with $\mathcal{H}_{\mathrm{TAI}}$. This service curve is shown below the S_k box in Figure B.17.

Now that we know the two service curves for S_k and $\mathcal{C}_\sigma$ as observed with $\mathcal{H}_{\mathrm{TAI}}$, the system S' on Figure B.17, that is the concatenation of S_k and $\mathcal{C}_\sigma$, offers to f the service curve $\beta^{\mathcal{H}_{\mathrm{TAI}}}_{S'} = \delta_{D^{\mathcal{H}_{\mathrm{TAI}}}_{f,S_k}} \otimes \delta_\eta \otimes \gamma_{Wr_0/\rho,b_0}$ when observed with $\mathcal{H}_{\mathrm{TAI}}$ [Le Boudec, Thiran 2001, Thm. 1.4.6]. Its shape is given in Figure B.18.

Conversely, the flow has, when observed with $\mathcal{H}_{\mathrm{TAI}}$ and at the input of S', both α_1 (PFR output arrival-curve property) and $\alpha_{2,k-1}$ (assumption in Lemma B.15) as arrival curves. Their shape are given in Figure B.18.

We apply the Network Calculus three-bound theorem [Le Boudec, Thiran 2001, Thm 1.4.2] to obtain a delay bound as the maximal horizontal distance between $\alpha_1 \otimes \alpha_2$ and β_{Hop_k}, reached at the location marked on Figure B.18. Note that knowing only α_1 does not prove the existence of a maximal horizontal distance because $Wr_0/\rho < \rho W r_0$ (for $\rho > 1$). However, knowing also $\alpha_{2,k-1}$ proves its existence because $Wr_0/\rho \geq \rho r_0$ because $W \geq \rho^2$. Geometrical considerations give that

$$D^{\mathcal{H}_{\mathrm{TAI}}}_{f,S_k+\mathtt{REG}_k} = D^{\mathcal{H}_{\mathrm{TAI}}}_{f,S_k} + \eta(1+\rho) + \frac{b_{2,(k-1)^*} - b_0 - \eta W r_0}{\rho r_0} \frac{\rho^2 - 1}{W - 1} \tag{B.80}$$

is a delay bound through S'.

S' is causal, lossless and FIFO for f because S_k and $\mathcal{C}_\sigma$ are. Hence it provides to f the service curve $\delta_{D^{\mathcal{H}_{\mathrm{TAI}}}_{f,S_k+\mathtt{REG}_k}}$ [Le Boudec, Thiran 2001, Prop. 1.3.3]. Both S' and the packetizer are causal, lossless and FIFO and the input of S' is pacektized. Hence their concatenation, *i.e.*, S'' offers to f the service curve $\delta_{D^{\mathcal{H}_{\mathrm{TAI}}}_{f,S_k+\mathtt{REG}_k}}$ to f [Le Boudec, Thiran 2001, Thm 1.7.1 Item 3] and is also causal, lossless and FIFO because both S' and P^L are.

Using one last time the Le Boudec, Thiran 2001, Prop. 1.3.3, this means that $\delta_{D^{\mathcal{H}_{\mathrm{TAI}}}_{f,S_k+\mathtt{REG}_k}}$ defined in (B.80) is a delay bound for f through S'', *i.e.*, through the concatenation of S_k and $\mathtt{REG}_k$.

(2) The system S'' in Figure B.17 offers to f the service curve $\delta_{D^{\mathcal{H}_{\mathrm{TAI}}}_{f,S_k+\mathtt{REG}_k}}$. We apply cite[Thm 1.4.3]leboudecNetworkCalculusTheory2001 with $\mathcal{H}_{\mathrm{TAI}}$ to obtain the result.

□

Proof of Proposition 5.7. (1) Applying Table 5.1 with $\mathcal{H}_g = \mathcal{H}_{\mathrm{TAI}}$ and $\mathcal{H}_i = \mathcal{H}_{\mathtt{REG}_0}$ proves that $\alpha_{2,0}$ is an arrival curve for the flow at its source when observed with $\mathcal{H}_{\mathrm{TAI}}$.

(2) and (3): The combination of the two statements is shown by induction on k. The base step $k = 0$ follows from the previous item. The induction step follows immediately from items (2) and (3) of Lemma B.15

□

B.3.8 Proof of Proposition 5.8

Proof of Proposition 5.8. Take the clock of the source to be exactly the TAI and let $\mathcal{H}_{\mathrm{Reg}} \neq \mathcal{H}_{\mathrm{TAI}}$ be the clock of the PFR. Since the PFR is non-adapted, it is configured with shaping curve $\sigma^{\mathcal{H}_{\mathrm{Reg}}} \triangleq \alpha^{\mathcal{H}_{\mathrm{TAI}}}_{f,\phi}$.

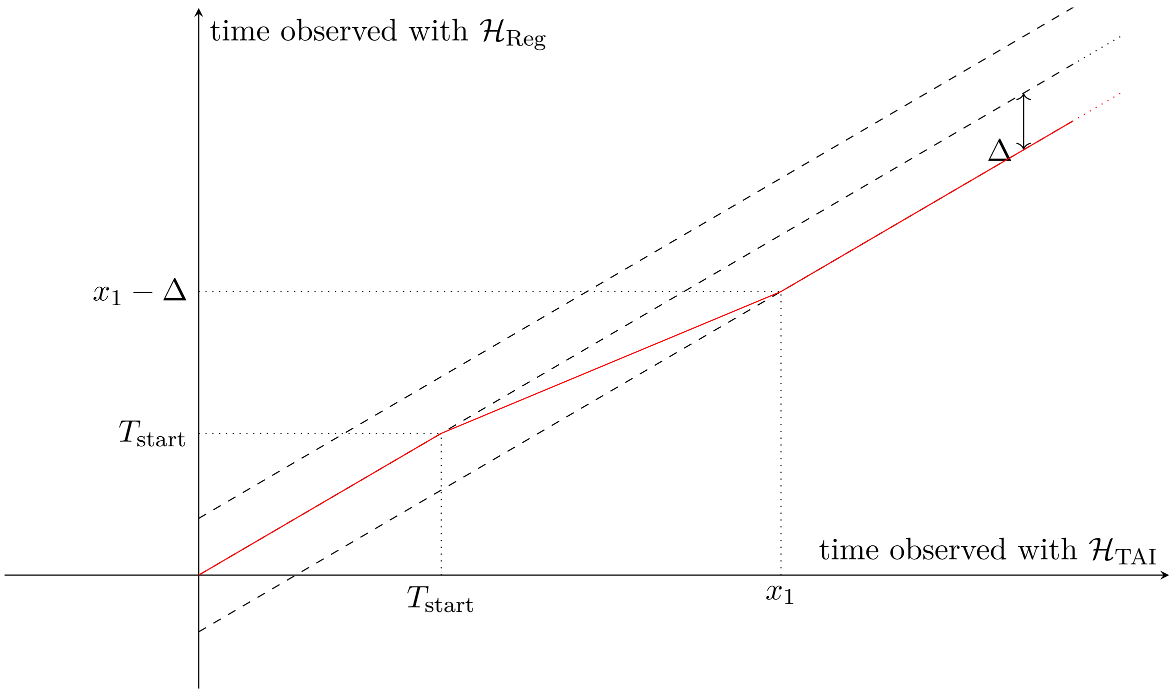


Figure B.19: Shape of the relative time function $d_{\text{PFR}\leftarrow\text{TAI}}$. When the TAI reaches T_{start}, the PFR clock $\mathcal{H}_{\text{Reg}}$ starts measuring the time slower (with a relative factor $1/\rho$) until its time is late by Δ compared to the TAI. After that point, it measures the time at the same speed but remain Δ seconds late.

Let $x_1 \triangleq T_{\text{start}} + \frac{\rho\Delta}{\rho-1}$ and take, for the adversarial relative time function $d_{\text{PFR}\leftarrow\text{TAI}}$, the following piecewise linear function

$$d_{\text{PFR}\leftarrow\text{TAI}}(t) \triangleq \begin{cases} t & \text{if } t \leq T_{\text{start}} \\ \frac{1}{\rho}(t - T_{\text{start}}) + T_{\text{start}} & \text{if } T_{\text{start}} < t \leq x_1 \\ t - \Delta & \text{if } x_1 < t \end{cases}$$

The shape of $d_{\text{PFR}\leftarrow\text{TAI}}$ is given in Figure B.19. It is continuous, strictly increasing, and meets the constraints of Equations (5.5) and (5.6).

As in Appendix B.3.5, let the source be greedy after T_{start} and generate packets of size ℓ bits, with $b \geq \ell$. Its cumulative arrival function at the source, measured in $\mathcal{H}_1$, is

$$R^{\mathcal{H}_{\text{TAI}}}(t) = \left\lfloor \frac{\alpha^{\mathcal{H}_{\text{TAI}}}(|t - T_{\text{start}}|^+)}{\ell} \right\rfloor \ell \tag{B.81}$$

where $\lfloor\cdot\rfloor$ is the floor function. The fact that this flow satisfies the arrival-curve constraint $\alpha^{\mathcal{H}_{\text{TAI}}}$ in $\mathcal{H}_{\text{TAI}}$ follows again from Le Boudec 2001, Thm III.2.

Consider now the first network element for the flow, S_1 in Figure 5.9, as having no delay, for any observation clock.

Let R' be the cumulative arrival function in the PFR, that is at the output of the network element. Then for any t, $R'^{\mathcal{H}_{\text{TAI}}}(t) = R^{\mathcal{H}_{\text{TAI}}}(t)$ and for any τ, $R'^{\mathcal{H}_{\text{Reg}}}(\tau) = R^{\mathcal{H}_{\text{Reg}}}(\tau)$.

With the same arguments as in Appendix B.3.5, we model the PFR as a fluid greedy shaper followed by a packetizer. The latter can be ignored for delay computations. The output of the fluid greedy shaper, observed with $\mathcal{H}_{\text{Reg}}$, is thus $R^{*\mathcal{H}_{\text{Reg}}} = R'^{\mathcal{H}_{\text{Reg}}} \otimes \gamma_{r,b}$. It follows that, for all τ, $R^{*\mathcal{H}_{\text{Reg}}}(\tau) \leq R^{\mathcal{H}_{\text{Reg}}}(T_{\text{start}}) + \alpha^{\mathcal{H}_{\text{TAI}}}(\tau - T_{\text{start}}) = r|\tau - T_{\text{start}}|^{+} + b$ by definition of T_{start}.

Now let x_2 be the next TAI time after x_1 at which the source finishes sending a packet. Then by definition of R and R'

$$R'^{\mathcal{H}_{\text{TAI}}}(x_2) = R^{\mathcal{H}_{\text{TAI}}}(x_2) = \alpha^{\mathcal{H}_{\text{TAI}}}(x_2 - T_{\text{start}}) \tag{B.82}$$

Now consider $x_2 + \Delta$ and compute the cumulative output of the regulator at $x_2 + \Delta$: $R^{*\mathcal{H}_{\text{TAI}}}(x_2 + \Delta) = R^{\mathcal{H}_{\text{Reg}}}(d_{\text{PFR}\leftarrow\text{TAI}}(x_2 + \Delta)) = R^{\mathcal{H}_{\text{Reg}}}(x_2)$ because $x_2 \geq x_1$ and by definition of $d_{\text{PFR}\leftarrow\text{TAI}}$.

Yet $R^{\mathcal{H}_{\text{Reg}}}(x_2) \leq \alpha^{\mathcal{H}_{\text{TAI}}}(x_2 - T_{\text{start}})$ so

$$R^{*\mathcal{H}_{\text{TAI}}}(x_2 + \Delta) \leq \alpha^{\mathcal{H}_{\text{TAI}}}(x_2 - T_{\text{start}}) \tag{B.83}$$

Combining Equations B.82 and B.83 proves

$$R^{*\mathcal{H}_{\text{TAI}}}(x_2 + \Delta) - R^{\mathcal{H}_{\text{TAI}}}(x_2) \leq 0 \tag{B.84}$$

Equation (B.84) proves that the delay of the packet output at x_2 (observed with $\mathcal{H}_{\text{TAI}}$) from the source exits the greedy shaper at $x_2 + \Delta$ (observed with $\mathcal{H}_{\text{TAI}}$). It has hence suffered a delay of Δ measured with $\mathcal{H}_{\text{TAI}}$, which is Δ more than the worst-case delay through the network element. The worst-case delay is hence lower-bounded by this reachable value. □

B.3.9 Proof of Proposition 5.9

Proof of Proposition 5.9. For each hop index k, the shaping curve of the non-adapted PFR $\texttt{REG}_k$ is a leaky-bucket curve σ (the arrival curve of the flow at its source, observed with the source's clock) that is concave and such that $\sigma(0^+) = b_0$, which is greater than the maxmimum packet length of the flow. In addition, the input of $\texttt{REG}_k$ is packetized. Thus $\texttt{REG}_k$ can be modeled as the concatenation of a fluid greedy shaper $\mathcal{C}_\sigma$ (that provides the non-adapted shaping curve σ as a service curve when observed with its own clock) and of a packetizer P^L that does not increase the hop delay.

For any hop index k, the PFR $\texttt{REG}_k$ is non-adapted, its configuration is

$$\begin{cases} r_{\texttt{REG}_k} = r_0 \\ b_{\texttt{REG}_k} = b_0 \end{cases}$$

One one hand, $\gamma_{r_{\texttt{REG}_k}, b_{\texttt{REG}_k}}$ is a service curve of $\mathcal{C}_\sigma$ when observed with its clock $\mathcal{H}_{\texttt{REG}_k}$. We use Proposition 5.4 and the synchronized part of Table 5.2 with $\mathcal{H}_g = \mathcal{H}_{\text{TAI}}$ and $\mathcal{H}_i = \mathcal{H}_{\texttt{REG}_k}$ and obtain that $\mathcal{C}_\sigma$ offers the service curve

$$\beta_{\mathcal{C}_\sigma}^{\mathcal{H}_{\text{TAI}}} = \left(\delta_\eta \otimes \gamma_{r_0/\rho, b_0}\right) \vee (\delta_{2\Delta} \otimes \gamma_{r_0, b_0}) \tag{B.85}$$

when observed with $\mathcal{H}_{\text{TAI}}$.

On the other hand, by definition of the PFR, $\gamma_{r_{\mathtt{REG}_{k-1}}, b_{\mathtt{REG}_{k-1}}}$ is an arrival curve of the flow at the output of the previous regulator, *i.e.*, at the input of S_k, when observed with $\mathcal{H}_{\mathtt{REG}_{k-1}}$. We apply Proposition 5.3 and the synchronized part of Table 5.1 with $\mathcal{H}_g = \mathcal{H}_{\text{TAI}}$ and $\mathcal{H}_i = \mathcal{H}_{\mathtt{REG}_{k-1}}$ and obtain that $\alpha^{\mathcal{H}_{\text{TAI}}}_{(k-1)*} = \gamma_{\rho r_0, b_0 + r_0\eta} \wedge \gamma_{r_0, b_0 + 2r_0\Delta}$ is an arrival curve of the flow at the input of hop k when observed with $\mathcal{H}_{\text{TAI}}$. Its shape is given in Figure B.20.

Assume now that $D^{\mathcal{H}_{\text{TAI}}}_{f,S_k}$ is a delay bound for f through S_k, computed by using the above $\alpha^{\mathcal{H}_{\text{TAI}}}_{(k-1)*}$ as an arrival curve in S_k when observed with $\mathcal{H}_{\text{TAI}}$. As S_k is causal, lossless and FIFO for f, S_k offers to f the service curve $\delta_{D^{\mathcal{H}_{\text{TAI}}}_{f,S_k}}$. The concatenation of S_k and $\mathcal{C}_\sigma$ (which we denoted as S' in Figure B.17) offers to f the service curve $\beta^{\mathcal{H}_{\text{TAI}}}_{S'} = \delta_{D^{\mathcal{H}_{\text{TAI}}}_{f,S_k}} \otimes \beta^{\mathcal{H}_{\text{TAI}}}_{\text{PFR}_k}$ when observed with $\mathcal{H}_{\text{TAI}}$. Its shape is given in Figure B.20.

We then compute the maximal horizontal distance in the figure. Geometrical considerations give

$$\begin{cases} y_A = b_0 + \dfrac{r_0}{\rho - 1}(2\Delta\rho - \eta) \\ y_B = b_0 + \dfrac{r_0}{\rho - 1}(2\Delta - \eta) \end{cases}$$

Hence, $y_A \geq y_B$ and the maximum horizontal distance is reached at A. We obtain

$$D^{\mathcal{H}_{\text{TAI}}}_{f,S_k+\mathtt{REG}_k} = D^{\mathcal{H}_{\text{TAI}}}_{f,S_k} + 4\Delta \tag{B.86}$$

And as the packetizer P^L does not increase the delay bound of the hop, the bound $D^{\mathcal{H}_{\text{TAI}}}_{f,S_k+\mathtt{REG}_k}$ obtained in (B.86) is indeed an upper-bound on the delay of f through the entire k-th hop.

□

B.3.10 Proof of Proposition 5.10

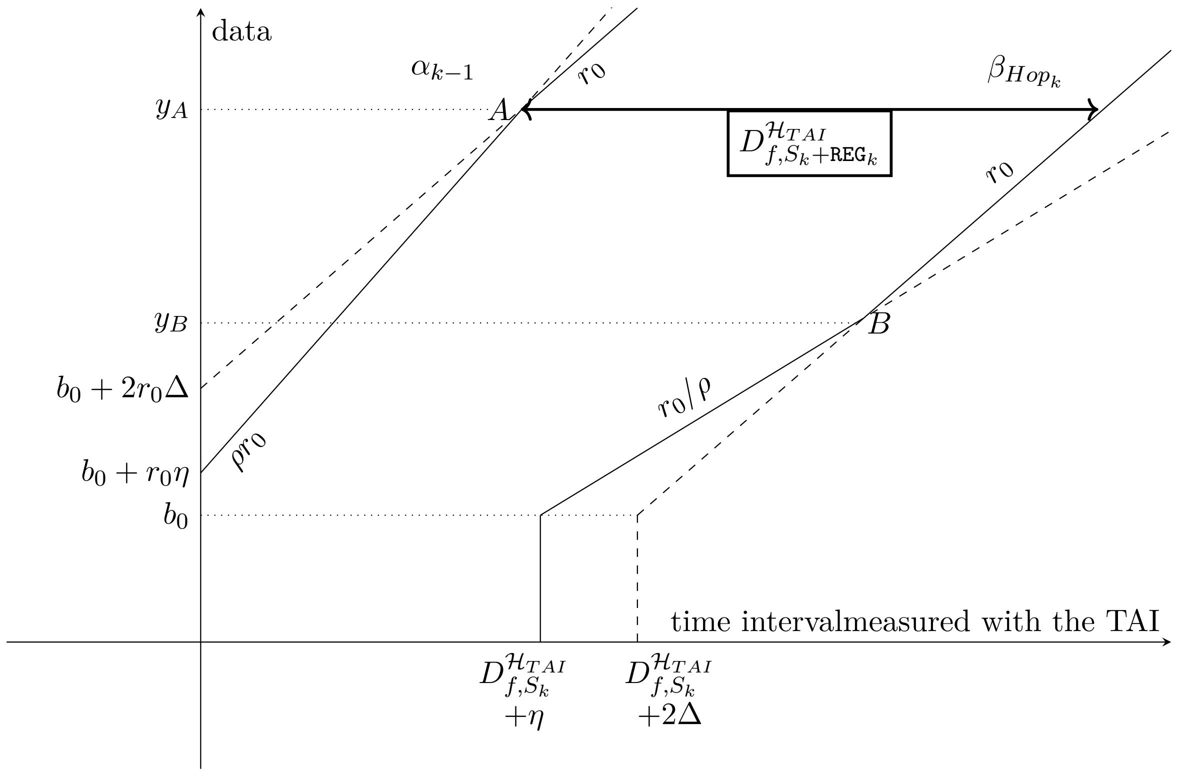
Proof of Proposition 5.10. Take $\eta \geq 0$, $\rho > 1$, $\Delta > 0$ and $n \geq 3$. The following example respects the constraints of the model and has unbounded flow latencies.

We consider n sources, each source generates a single flow to the FIFO system. W e choose a FIFO system with infinite service and with $\mathcal{H}_{\text{FIFO}} = \mathcal{H}_{\text{IR}}$ such that the delay of the flows trough the FIFO system, when observed in the clock $\mathcal{H}_{\text{IR}}$, equals zero. We further take $\mathcal{H}_{\text{IR}} = \mathcal{H}_{\text{TAI}}$. If the TAI delay is not bounded, then it is also not bounded in any other clock that meets the stability requirements of Equation (5.5).

Adversarial Clocks: We consider a starting point $x_1 \geq T_{\text{start}}$. We choose a slope s_1 such that $1 < s_1 \leq \min(1.5, \sqrt{\rho})$ and define

$$I \triangleq \frac{\Delta s_1}{s_1 - 1}$$

We also consider any $\epsilon > 0$ such that $\epsilon < I(1 - \frac{1}{s_1})$. ϵ is well defined because $s_1 > 1$. We also define $\tau \triangleq nI/s_1 + n\epsilon$ and $x_j \triangleq x_1 + (j-1)I/s_1 + (j-1)\epsilon$ for $j = 1 \ldots n$. Now, for every



data
α_{k-1}
r_0
β_{Hop_k}
y_A
A
$D^{\mathcal{H}_{TAI}}_{f,S_k+\mathtt{REG}_k}$
r_0
y_B
B
$b_0 + 2r_0\Delta$
r_0/ρ
ρr_0
$b_0 + r_0\eta$
b_0
time intervalmeasured with the TAI
$D^{\mathcal{H}_{TAI}}_{f,S_k}$ $+\eta$
$D^{\mathcal{H}_{TAI}}_{f,S_k}$ $+2\Delta$

Figure B.20: Delay-bound computation for the flow through the concatenation of S_k with the fluid greedy shaper $\mathcal{C}_\sigma$ when the regulator is a PFR and the network is synchronized. This also gives a delay bound for the entire k-th hop as the packetizer in the PFR model does not increase the hop delay bound.

clock j, we choose as relative time function $d_{j\leftarrow\text{IR}}$ the following piecewise linear function

$$d_j = d_{j\leftarrow\text{IR}} : t \mapsto \begin{cases} t - \Delta/2 & \text{if } t \leq x_j \\ s_1(t - x_j) + x_j - \Delta/2 & \text{if } x_j < t \leq x_j + I/s_1 \\ \left.\begin{matrix} 1/s_1(t - x_j + I/s_1) \\ +I + x_j - \Delta/2 \end{matrix}\right\} & \text{if } x_j + \dfrac{I}{s_1} < t \leq x_j + \dfrac{I}{s_1} + I \\ t - \Delta/2 & \text{if } x_j + \dfrac{I}{s_1} + I < t \leq x_j + \tau \\ \tau + d_j(t - \tau) & \text{if } x_j + \tau < t \end{cases}$$

The shape of function $d_{j\leftarrow\text{IR}}$ is available in Figure B.21. We obtain directly the following properties for any j

- $d_{j\leftarrow\text{IR}}$ is continuous and strictly increasing
- The time-error function $t \mapsto d_{j\leftarrow\text{IR}}(t) - t$ is periodic with period τ for $t \geq x_j$

Also, for any j, j' $d_{j'\leftarrow j}(t) = d_{j'\leftarrow\text{IR}}(d^{-1}_{j\leftarrow\text{IR}}(t))$. As $s_1 \leq \sqrt{\rho}$ and $|d_{j\leftarrow\text{IR}}(t) - t| \leq \Delta/2$, any pair of clocks $(\mathcal{H}_j, \mathcal{H}_{j'})$ meets the constraints of Equations (5.5) and (5.6), which shows that our adversarial clocks are within the synchronized time model proposed in Section 5.2.1.

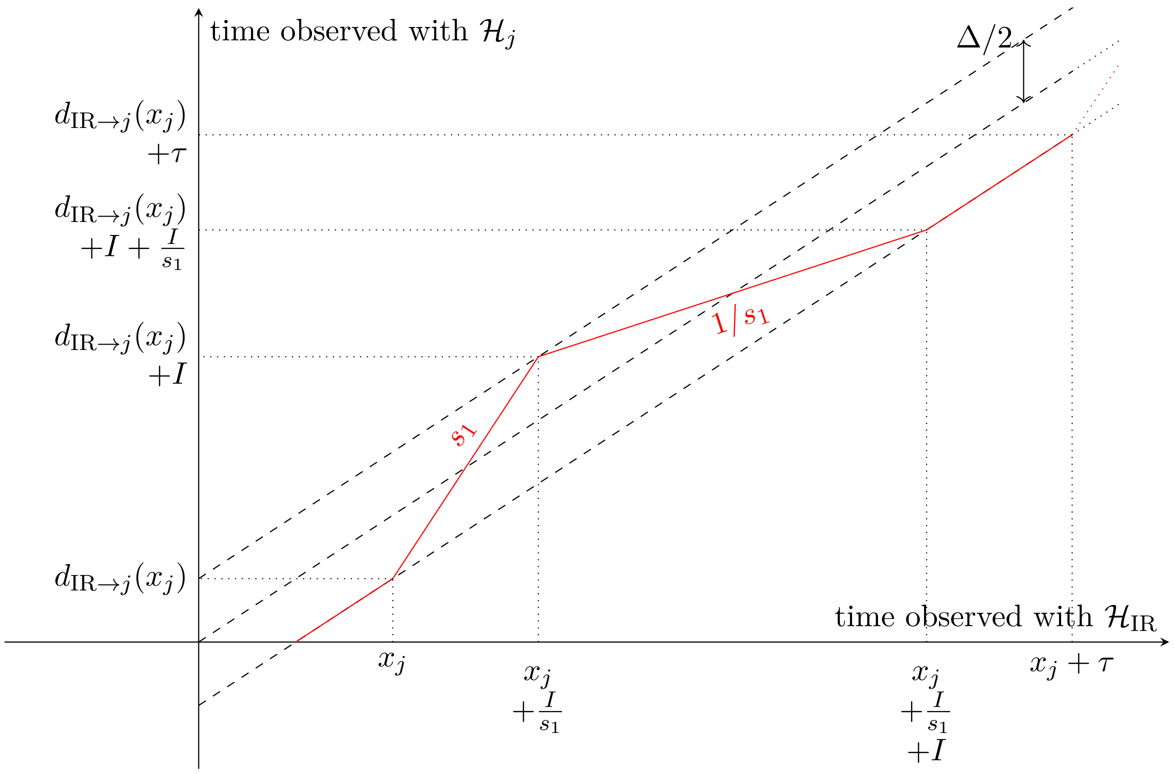

time observed with $\mathcal{H}_j$
$\Delta/2$
$d_{\mathrm{IR}\to j}(x_j)$ $+\tau$
$d_{\mathrm{IR}\to j}(x_j)$ $+I+\frac{I}{s_1}$
$d_{\mathrm{IR}\to j}(x_j)$ $+I$
$d_{\mathrm{IR}\to j}(x_j)$
$1/s_1$
s_1
time observed with $\mathcal{H}_{\mathrm{IR}}$
x_j
x_j $+\frac{I}{s_1}$
x_j $+\frac{I}{s_1}$ $+I$
$x_j+\tau$

Figure B.21: Shape of the time function $d_{j\leftarrow \mathrm{IR}}$. The shape is periodic, with period τ. When clock $\mathcal{H}_{\mathrm{IR}}$ reaches x_j, clock $\mathcal{H}_j$ starts counting the time faster (with a relative factor s_1) until $\mathcal{H}_{\mathrm{IR}}$ counts I/s_1 more seconds. Then, $\mathcal{H}_j$ counts the time slower (with a relative factor $1/s_1$). When the time-error function between $\mathcal{H}_{\mathrm{IR}}$ and $\mathcal{H}_j$ reaches $-\Delta/2$, $\mathcal{H}_j$ counts the time at the same speed as $\mathcal{H}_{\mathrm{IR}}$ until the next period starts.

Adversarial Traffic Generation: Let l be any arbitrary data size. Each source j is configured to send a packet of size l when its local clock reaches $d_j(x_j)+k\tau$ and $d_j(x_j)+k\tau+I$ for all $k\in\mathbb{N}$. Figure B.22 presents the traffic generation of source j within one period, observed with its internal clock $\mathcal{H}_j$. When observed with $\mathcal{H}_j$, the traffic generation is periodic of period τ.

As $n\geq 3$ and $s_1<1.5$, $\tau\geq 2I+2\epsilon\geq 2I$ and $\tau-I\geq I$. Hence, the minimum duration between two packets generated by source j, measured with $\mathcal{H}_j$ is I. This proves that each flow exits its respective source j with a leaky-bucket arrival curve $\gamma_{\frac{l}{I},l}$ (rate l/I, burst l) when observed using $\mathcal{H}_j$. We now assume that the interleaved regulator is configured with the same leaky-bucket arrival curve $\gamma_{\frac{l}{I},l}$ for all the flows.

Figure B.23 presents the timeline of packets generated by source j but as observed with $\mathcal{H}_{\mathrm{IR}}$. The IR has to regulate the same timeline for all the n inputs, based on its configuration. For $j=1\ldots n$ and for $k\in\mathbb{N}$ we note $A^1_{j,k}=x_j+k\tau$ the arrival time in the IR of the first packet of the the kth period of source j measured with $\mathcal{H}_{\mathrm{IR}}$ and $A^2_{j,k}=x_j+k\tau+\frac{I}{s_1}$ the arrival time of the second packet, still measured with $\mathcal{H}_{\mathrm{IR}}$. Also, note $D^1_{j,k}$ and $D^2_{j,k}$ their respective release time out of the IR, again measured using $\mathcal{H}_{\mathrm{IR}}$.

Figure B.24 presents the arrival and release times of the packets for two consecutive sources. Assume for instance that the sources have been idle for a while, then $D^1_{j,k}=A^1_{j,k}$.

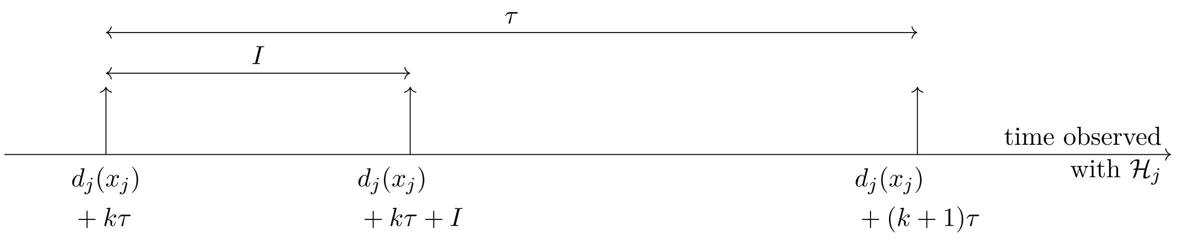


Figure B.22: Generation of packets as observed with $\mathcal{H}_j$. The traffic profile is periodic of period τ. Source j sends a packet when the internal clock reaches $d_j(x_j) + k\tau$ for some $k \in \mathbb{N}$, then it sends another packet after a duration of I counted using $\mathcal{H}_j$, and finally restarts at the next period.

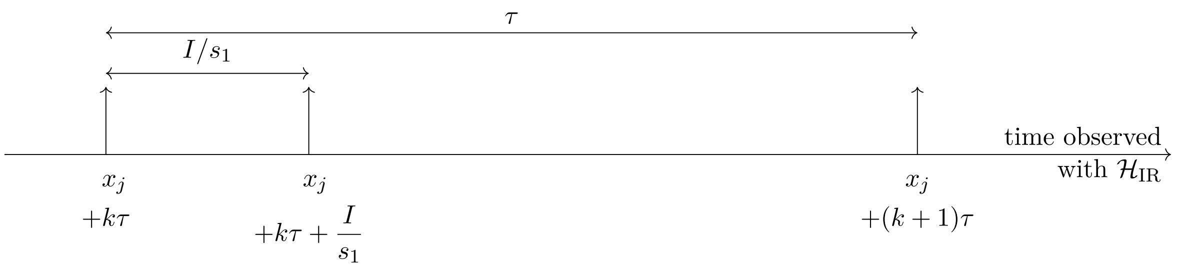

τ
I/s_1
time observed with $\mathcal{H}_{\text{IR}}$
$x_j + k\tau$
$x_j + k\tau + \frac{I}{s_1}$
$x_j + (k+1)\tau$

Figure B.23: Generation of packets as observed with $\mathcal{H}_{\text{IR}}$. The traffic profile is periodic of period τ. When observing with $\mathcal{H}_{\text{IR}}$, source j sends packets at $x_j + k\tau$ and $x_j + k\tau + \frac{I}{s_1}$ for all $k \in \mathbb{N}$.

The instant $A^2_{j,k}$ is, from the perspective of $\mathcal{H}_{\text{IR}}$, too soon by $I(1-\frac{1}{s_1})$. Using the IR equations [Le Boudec 2018], the IR has to delay the packet and

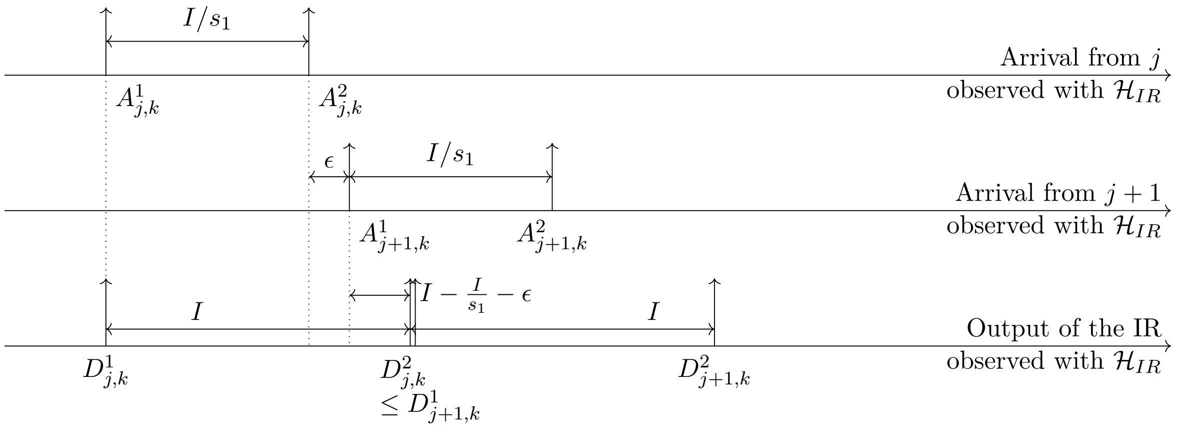
$$\forall j = 1 \dots n, \forall k \in \mathbb{N}, \quad D^2_{j,k} \geq D^1_{j,k} + I \tag{B.87}$$

As $x_{j+1} = x_j + \frac{I}{s_1} + \epsilon$, packet $A^1_{j+1,k}$ arrives ϵ seconds after $A^2_{j,k}$ (measured using $\mathcal{H}_{\text{IR}}$). As $\epsilon < I(1 - \frac{1}{s_1})$, the packet at $A^1_{j+1,k}$ arrives before the previous packet of the previous source could be released out of the IR. Because the IR only looks at the head-of-line packet, and using the IR equations, we obtain

$$\forall j = 1 \dots (n-1), \forall k \in \mathbb{N}, \quad D^1_{j+1,k} \geq D^2_{j,k} \tag{B.88}$$

Combining Equations (B.87) and (B.88) gives, by induction,

$$D^2_{n,k} \geq D^1_{1,k} + nI \tag{B.89}$$



I/s_1
Arrival from j observed with $\mathcal{H}_{IR}$
$A^1_{j,k}$
$A^2_{j,k}$
ϵ
I/s_1
Arrival from $j+1$ observed with $\mathcal{H}_{IR}$
$A^1_{j+1,k}$
$A^2_{j+1,k}$
I
$I - \frac{I}{s_1} - \epsilon$
I
Output of the IR observed with $\mathcal{H}_{IR}$
$D^1_{j,k}$
$D^2_{j,k} \leq D^1_{j+1,k}$
$D^2_{j+1,k}$

Figure B.24: Traffic arrival from two successive upstream sources, as observed with $\mathcal{H}_{\text{IR}}$ and release time of the packets, again observed with $\mathcal{H}_{\text{IR}}$.

Now we can note that

$$\begin{aligned}
A^1_{1,k+1} &= x_1 + k\tau + \tau \\
&= x_1 + k\tau + n\frac{I}{s_1} + n\epsilon \\
&= x_1 + (n-1)\frac{I}{s_1} + (n-1)\epsilon + k\tau + \frac{I}{s_1} + \epsilon \\
&= x_n + k\tau + \frac{I}{s_1} + \epsilon \\
&= A^2_{n,k} + \epsilon
\end{aligned}$$

Hence, the first packet of the $(k+1)$th period of the first upstream source arrives ϵ seconds (counted with $\mathcal{H}_{\text{IR}}$) after the second packet of the kth period of the last source, so we also have

$$D^1_{1,k+1} \geq D^2_{n,k} \tag{B.90}$$

Combining Equations (B.87) and (B.90) gives

$$D^1_{1,k} \geq D^1_{1,1} + (k-1)nI = x_1 + (k-1)nI \tag{B.91}$$

Because we have $D^1_{1,1} = x_1$, as the network was empty before. The delay suffered through the IR by the first packet of the kth period of the first source is, when measured with $\mathcal{H}_{\text{IR}}$

$$\begin{aligned}
D^1_{1,k} - A^1_{1,k} &\geq x_1 + (k-1)nI - x_1 - (k-1)\tau && \text{(B.92)}\\
&\geq x_1 + (k-1)nI - x_1 - (k-1)n\frac{I}{s_1} - (k-1)n\epsilon && \text{(B.93)}\\
&\geq (k-1)n\left(I\left(1 - \frac{1}{s_1}\right) - \epsilon\right) && \text{(B.94)}
\end{aligned}$$

As we have arbitrary selected ϵ such that $\epsilon < I(1 - \frac{1}{s_1})$, we obtain $I(1 - \frac{1}{s_1}) - \epsilon > 0$ thus the above delay lower-bound diverges as k increases, so the delay through the IR is unbounded when seen from $\mathcal{H}_{\mathrm{IR}}$, which proves the instability.

Remark: Equation (B.94) proves that at each period of duration τ, the delay increases by $nI(1 - \frac{1}{s_1}) - n\epsilon$. The divergence of the delay per second is

$$\mathrm{div} = \frac{nI(1 - \frac{1}{s_1}) - n\epsilon}{\frac{nI}{s_1} + n\epsilon}$$

This divergence is valid for any $\epsilon > 0$, with $\epsilon < I(1 - \frac{1}{s_1})$. Taking $\epsilon \to 0$, the divergence can be as large as

$$\begin{aligned}\lim_{\epsilon \to 0} \mathrm{div} &= \frac{nI(1 - \frac{1}{s_1})}{\frac{nI}{s_1}} \\ &= s_1 - 1\end{aligned}$$

s_1 can be as large as $\sqrt{\rho}$, so the divergence of the delay can be as large as $\sqrt{\rho} - 1$, for any $n \geq 3, \Delta > 0$.

$\square$

Bibliography

[Aguirre Rodrigo 2020] Aguirre Rodrigo, Guillermo (2020). *Simulation of Instability in Time-Sensitive Networks with Regulators and Imperfect Clocks.* EPFL/LCA2. 80 pp. URL: https://infoscience.epfl.ch/record/294616.

[Amari, Mifdaoui 2017] Amari, A. and A. Mifdaoui (Aug. 2017). "Worst-Case Timing Analysis of Ring Networks with Cyclic Dependencies Using Network Calculus." In: *2017 IEEE 23rd International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA).* 2017 IEEE 23rd International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), pp. 1–10. DOI: 10.1109/RTCSA.2017.8046319.

[Andrews 2009] Andrews, Matthew (July 2009). "Instability of FIFO in the Permanent Sessions Model at Arbitrarily Small Network Loads." In: *ACM Trans. Algorithms* 5.3, 33:1–33:29. ISSN: 1549-6325. DOI: 10.1145/1541885.1541894. URL: http://doi.acm.org/10.1145/1541885.1541894 (visited on 04/10/2019).

[Baharev, Schichl, Neumaier 2015] Baharev, Ali, Hermann Schichl, and Arnold Neumaier (2015). "An Exact Method for the Minimum Feedback Arc Set Problem." In: Fakultät für Mathematik, Universität Wien. URL: https://www.mat.univie.ac.at/~neum/papers.html.

[Bauer, Scharbarg, Fraboul 2010] Bauer, Henri, Jean-Luc Scharbarg, and Christian Fraboul (Nov. 2010). "Applying Trajectory Approach with Static Priority Queuing for Improving the Use of Available AFDX Resources." In: *18th International Conference on Real-Time and Network Systems.* Toulouse, France, pp. 69–78. URL: https://hal.archives-ouvertes.fr/hal-00544508 (visited on 06/27/2022).

[Bisti, Lenzini, Mingozzi, *et al.* 2008] Bisti, Luca, Luciano Lenzini, Enzo Mingozzi, and Giovanni Stea (Oct. 20, 2008). "Estimating the Worst-Case Delay in FIFO Tandems Using Network Calculus." In: *Proceedings of the 3rd International Conference on Performance Evaluation Methodologies and Tools.* ValueTools '08. Brussels, BEL: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), pp. 1–10. ISBN: 978-963-9799-31-8. DOI: 10.4108/ICST.VALUETOOLS2008.4388. URL: https://doi.org/10.4108/ICST.VALUETOOLS2008.4388 (visited on 05/09/2022).

[Bisti, Lenzini, Mingozzi, *et al.* 2010] – (2010). "DEBORAH: A Tool for Worst-Case Analysis of FIFO Tandems." In: *Leveraging Applications of Formal Methods, Verification, and Validation.* Ed. by Tiziana Margaria and Bernhard Steffen. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 152–168. ISBN: 978-3-642-16558-0.

[Bondorf 2017] Bondorf, Steffen (May 2017). "Better Bounds by Worse Assumptions — Improving Network Calculus Accuracy by Adding Pessimism to the Network Model." In: *2017 IEEE International Conference on Communications (ICC).* 2017 IEEE International Conference on Communications (ICC), pp. 1–7. DOI: 10.1109/ICC.2017.7996996.

[Bondorf, Nikolaus, Schmitt 2017] Bondorf, Steffen, Paul Nikolaus, and Jens B. Schmitt (June 2017). "Quality and Cost of Deterministic Network Calculus: Design and Evaluation of an Accurate and Fast Analysis." In: *Proc. ACM Meas. Anal. Comput. Syst.* 1.1. +1 poster dans ACM SIGMETRICS 2017,
"On the Capacity Requirement for Arbitrary End-to-End Deadline and Reliability Guarantees in Multi-hop Networks"

Han Deng,I-Hong Hou (Texas A&M University)
+1 poster dans SIGMETRICS 2016:
"Network Calculus Analysis of a Feedback System with Random Service"
Alireza Shekaramiz, Jorg Liebeherr, Almut Burchard (University of Toronto), 16:1–16:34. ISSN: 2476-1249. DOI: 10.1145/3084453. URL: http://doi.acm.org/10.1145/3084453 (visited on 10/07/2019).

[Bondorf, Schmitt 2016a] Bondorf, Steffen and Jens Schmitt (Jan. 4, 2016a). "Calculating Accurate End-to-End Delay Bounds - You Better Know Your Cross-Traffic." In: *Proceedings of the 9th EAI International Conference on Performance Evaluation Methodologies and Tools*. VALUETOOLS'15. Brussels, BEL: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), pp. 17–24. ISBN: 978-1-63190-096-9. DOI: 10.4108/eai.14-12-2015.2262565. URL: https://doi.org/10.4108/eai.14-12-2015.2262565 (visited on 06/08/2022).

[Bondorf, Schmitt 2016b] – (2016b). "Improving Cross-Traffic Bounds in Feed-Forward Networks – There Is a Job for Everyone." In: *Measurement, Modelling and Evaluation of Dependable Computer and Communication Systems*. Ed. by Anne Remke and Boudewijn R. Haverkort. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 9–24. ISBN: 978-3-319-31559-1. DOI: 10.1007/978-3-319-31559-1_3.

[Bondorf, Schmitt 2014] Bondorf, Steffen and Jens B. Schmitt (Dec. 9, 2014). "The DiscoDNC v2: A Comprehensive Tool for Deterministic Network Calculus." In: *Proceedings of the 8th International Conference on Performance Evaluation Methodologies and Tools*. VALUETOOLS '14. Brussels, BEL: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), pp. 44–49. ISBN: 978-1-63190-057-0. DOI: 10.4108/icst.Valuetools.2014.258167. URL: https://doi.org/10.4108/icst.Valuetools.2014.258167 (visited on 06/08/2022).

[Bouillard 2021] Bouillard, Anne (June 2021). "Individual Service Curves for Bandwidth-Sharing Policies Using Network Calculus." In: *IEEE Networking Letters* 3.2, pp. 80–83. ISSN: 2576-3156. DOI: 10.1109/LNET.2021.3067766.

[Bouillard 2022] – (Feb. 1, 2022). "Trade-off between Accuracy and Tractability of Network Calculus in FIFO Networks." In: *Performance Evaluation* 153, p. 102250. ISSN: 0166-5316. DOI: 10.1016/j.peva.2021.102250. URL: https://www.sciencedirect.com/science/article/pii/S0166531621000675 (visited on 05/10/2022).

[Bouillard, Boyer, Le Corronc 2018] Bouillard, Anne, Marc Boyer, and Euriell Le Corronc (2018). *Deterministic Network Calculus: From Theory to Practical Implementation*. Networks and Telecommunications. Wiley. ISBN: 978-1-84821-852-9. URL: http://doi.org/10.1002/9781119440284.

[Bouillard, Jouhet, Thierry 2010] Bouillard, Anne, Laurent Jouhet, and Eric Thierry (Mar. 2010). "Tight Performance Bounds in the Worst-Case Analysis of Feed-Forward Networks." In: *2010 Proceedings IEEE INFOCOM*. 2010 Proceedings IEEE INFOCOM, pp. 1–9. DOI: 10.1109/INFCOM.2010.5461912.

[Bouillard, Nowak 2015] Bouillard, Anne and Thomas Nowak (Sept. 1, 2015). "Fast Symbolic Computation of the Worst-Case Delay in Tandem Networks and Applications." In: *Performance Evaluation*. Special Issue: Performance 2015 91, pp. 270–285. ISSN: 0166-5316.

DOI: 10.1016/j.peva.2015.06.016. URL: http://www.sciencedirect.com/science/article/pii/S0166531615000644 (visited on 06/24/2020).

[Bouillard, Stea 2012] Bouillard, Anne and Giovanni Stea (Oct. 2012). “Exact Worst-Case Delay for FIFO-multiplexing Tandems.” In: *6th International ICST Conference on Performance Evaluation Methodologies and Tools*. 6th International ICST Conference on Performance Evaluation Methodologies and Tools, pp. 158–167. DOI: 10.4108/valuetools.2012.250090.

[Bouillard, Stea 2015] – (Oct. 2015). “Exact Worst-Case Delay in FIFO-Multiplexing Feed-Forward Networks.” In: *IEEE/ACM Transactions on Networking* 23.5, pp. 1387–1400. ISSN: 1558-2566. DOI: 10.1109/TNET.2014.2332071.

[Boyer, Dufour, Santinelli 2013] Boyer, Marc, Guillaume Dufour, and Luca Santinelli (Oct. 16, 2013). “Continuity for Network Calculus.” In: *Proceedings of the 21st International Conference on Real-Time Networks and Systems*. RTNS ’13. New York, NY, USA: Association for Computing Machinery, pp. 235–244. ISBN: 978-1-4503-2058-0. DOI: 10.1145/2516821.2516840. URL: https://doi.org/10.1145/2516821.2516840 (visited on 01/16/2022).

[Boyer, Navet, Olive, *et al.* 2010] Boyer, Marc, Nicolas Navet, Xavier Olive, and Eric Thierry (2010). “The PEGASE Project: Precise and Scalable Temporal Analysis for Aerospace Communication Systems with Network Calculus.” In: *Leveraging Applications of Formal Methods, Verification, and Validation*. Ed. by Tiziana Margaria and Bernhard Steffen. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, pp. 122–136. ISBN: 978-3-642-16558-0. DOI: 10.1007/978-3-642-16558-0_13.

[Boyer, Stea, Sofack 2012] Boyer, Marc, Giovanni Stea, and William Mangoua Sofack (Oct. 2012). “Deficit Round Robin with Network Calculus.” In: *6th International ICST Conference on Performance Evaluation Methodologies and Tools*. 6th International ICST Conference on Performance Evaluation Methodologies and Tools, pp. 138–147. DOI: 10.4108/valuetools.2012.250202.

[Bramson 1996] Bramson, Maury (Mar. 1, 1996). “Convergence to Equilibria for Fluid Models of Head-of-the-Line Proportional Processor Sharing Queueing Networks.” In: *Queueing Systems* 23.1, pp. 1–26. ISSN: 1572-9443. DOI: 10.1007/BF01206549. URL: https://doi.org/10.1007/BF01206549 (visited on 04/27/2022).

[Chang 1997] Chang, Cheng-Shang (Apr. 1997). “A Filtering Theory for Deterministic Traffic Regulation.” In: *Proceedings of INFOCOM ’97*. Proceedings of INFOCOM ’97. Vol. 2, 436–443 vol.2. DOI: 10.1109/INFCOM.1997.644492.

[Chang 2000] – (2000). *Performance Guarantees in Communication Networks*. Springer London. DOI: 10.1007/978-1-4471-0459-9. URL: https://doi.org/10.1007/978-1-4471-0459-9.

[Charny, Le Boudec 2000] Charny, Anna and Jean-Yves Le Boudec (2000). “Delay Bounds in a Network with Aggregate Scheduling.” In: *Quality of Future Internet Services*. Ed. by Jon Crowcroft, James Roberts, and Mikhail I. Smirnov. Lecture Notes in Computer Science. Springer Berlin Heidelberg, pp. 1–13. ISBN: 978-3-540-39939-1. URL: https://link.springer.com/chapter/10.1007/3-540-39939-9_1.

[Chlamtac, Farago, Zhang, *et al.* 1998] Chlamtac, I., A. Farago, Hongbiao Zhang, and A. Fumagalli (Aug. 1998). “A Deterministic Approach to the End-to-End Analysis of Packet

Flows in Connection-Oriented Networks.” In: *IEEE/ACM Transactions on Networking* 6.4, pp. 422–431. ISSN: 1558-2566. DOI: 10.1109/90.720877.

[Ciucu, Schmitt, Wang 2011] Ciucu, Florin, Jens Schmitt, and Hao Wang (Apr. 2011). “On Expressing Networks with Flow Transformations in Convolution-Form.” In: *2011 Proceedings IEEE INFOCOM*. 2011 Proceedings IEEE INFOCOM, pp. 1979–1987. DOI: 10.1109/INFCOM.2011.5935003.

[Clarke, Emerson, Sistla 1986] Clarke, E. M., E. A. Emerson, and A. P. Sistla (Apr. 1, 1986). “Automatic Verification of Finite-State Concurrent Systems Using Temporal Logic Specifications.” In: *ACM Transactions on Programming Languages and Systems* 8.2, pp. 244–263. ISSN: 0164-0925. DOI: 10.1145/5397.5399. URL: https://doi.org/10.1145/5397.5399 (visited on 05/29/2022).

[Coudron, Secci 2015] Coudron, Matthieu and Stefano Secci (2015). “Per Node Clocks to Simulate Time Desynchronization in Networks.” In: URL: https://www.nsnam.org/workshops/wns3-2016/posters/per-node-clock-abstract.pdf.

[Cruz 1991a] Cruz, Rene L. (Jan. 1991a). “A Calculus for Network Delay. I. Network Elements in Isolation.” In: *IEEE Transactions on Information Theory* 37.1, pp. 114–131. ISSN: 1557-9654. DOI: 10.1109/18.61109.

[Cruz 1991b] – (Jan. 1991b). “A Calculus for Network Delay. II. Network Analysis.” In: *IEEE Transactions on Information Theory* 37.1, pp. 132–141. ISSN: 1557-9654. DOI: 10.1109/18.61110.

[Cuenot 2021] Cuenot, Philippe (Sept. 24, 2021). “TSN for Critical Embedded Systems.” École d’Été Temps Réel (Poitiers, France). URL: https://etr2021.ensma.fr/files/p-cuenot-tsn-irt.pdf (visited on 05/26/2022).

[Daigmorte 2019] Daigmorte, Hugo (Jan. 21, 2019). “Analyse Des Interactions Entre Flux Synchrones et Flux Asynchrones Dans Les Réseaux Temps Réel.” Sous la direction de Marc Boyer. Soutenue le 21-01-2019,à Toulouse, ISAE , dans le cadre de École doctorale Mathématiques, informatique et télécommunications (Toulouse) , en partenariat avec Équipe d’accueil doctoral Modélisation et ingénierie des systèmes (Toulouse, Haute-Garonne) (équipe de recherche) et de Office national d’études et recherches aérospatiales. Département Traitement de l’Information et Systèmes (DTIS) (laboratoire) . These de doctorat. Toulouse, ISAE. URL: http://theses.fr/2019ESAE0003 (visited on 01/21/2022).

[Daigmorte, Boyer 2016] Daigmorte, Hugo and Marc Boyer (Oct. 19, 2016). “Traversal Time for Weakly Synchronized CAN Bus.” In: *Proceedings of the 24th International Conference on Real-Time Networks and Systems*. RTNS ’16. Brest, France: Association for Computing Machinery, pp. 35–44. ISBN: 978-1-4503-4787-7. DOI: 10.1145/2997465.2997477. URL: https://doi.org/10.1145/2997465.2997477 (visited on 01/14/2020).

[Daigmorte, Boyer 2017] – (Oct. 4, 2017). “Evaluation of Admissible CAN Bus Load with Weak Synchronization Mechanism.” In: *Proceedings of the 25th International Conference on Real-Time Networks and Systems*. RTNS ’17. Grenoble, France: Association for Computing Machinery, pp. 277–286. ISBN: 978-1-4503-5286-4. DOI: 10.1145/3139258.3139261. URL: https://doi.org/10.1145/3139258.3139261 (visited on 01/14/2020).

[Daigmorte, Boyer, Migge 2017] Daigmorte, Hugo, Marc Boyer, and Jörn Migge (2017). “Reducing CAN Latencies by Use of Weak Synchronization between Stations.” In.

[Daigmorte, Boyer, Zhao 2018] Daigmorte, Hugo, Marc Boyer, and Luxi Zhao (June 2018). "Modelling in Network Calculus a TSN Architecture Mixing Time-Triggered, Credit Based Shaper and Best-Effort Queues." working paper or preprint. URL: https://hal.archives-ouvertes.fr/hal-01814211 (visited on 04/28/2022).

[Dal Fabbro 2020] Dal Fabbro, Nicolò (2020). *Stability of Time-Sensitive Networks: Strategies for Partial Regulation.* 11 pp. URL: https://infoscience.epfl.ch/record/294617.

[Dang, Mifdaoui 2014] Dang, Dinh-Khanh and Ahlem Mifdaoui (Sept. 2014). "Timing Analysis of TDMA-Based Networks Using Network Calculus and Integer Linear Programming." In: *2014 IEEE 22nd International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems.* 2014 IEEE 22nd International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems, pp. 21–30. DOI: 10.1109/MASCOTS.2014.12.

[Dierikx, Wallin, Fordell, *et al.* 2016] Dierikx, E. F., A. E. Wallin, T. Fordell, J. Myyry, P. Koponen, M. Merimaa, T. J. Pinkert, J. C. J. Koelemeij, H. Z. Peek, and R. Smets (July 2016). "White Rabbit Precision Time Protocol on Long-Distance Fiber Links." In: *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control* 63.7, pp. 945–952. DOI: 10.1109/TUFFC.2016.2518122.

[Docquier, Song, Chevrier, *et al.* 2020] Docquier, Théo, Ye-Qiong Song, Vincent Chevrier, Ludovic Pontnau, and Abdelaziz Ahmed-Nacer (Nov. 2020). "Determining a Tight Worst-Case Delay of Switched Ethernet Network in IEC 61850 Architectures." In: *2020 IEEE 45th Conference on Local Computer Networks (LCN).* 2020 IEEE 45th Conference on Local Computer Networks (LCN), pp. 184–194. DOI: 10.1109/LCN48667.2020.9314800.

[Eles, Doboli, Pop, *et al.* 2000] Eles, P., A. Doboli, P. Pop, and Z. Peng (Oct. 2000). "Scheduling with Bus Access Optimization for Distributed Embedded Systems." In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 8.5, pp. 472–491. ISSN: 1557-9999. DOI: 10.1109/92.894152.

[Falk, Hellmanns, Carabelli, *et al.* 2019] Falk, Jonathan, David Hellmanns, Ben Carabelli, Naresh Nayak, Frank Dürr, Stephan Kehrer, and Kurt Rothermel (Mar. 2019). "NeSTiNg: Simulating IEEE Time-sensitive Networking (TSN) in OMNeT++." In: *2019 International Conference on Networked Systems (NetSys).* 2019 International Conference on Networked Systems (NetSys), pp. 1–8. DOI: 10.1109/NetSys.2019.8854500.

[Farkas 2018] Farkas, János (Nov. 11, 2018). "TSN Basic Concepts." URL: https://www.ieee802.org/1/files/public/docs2018/detnet-tsn-farkas-tsn-basic-concepts-1118-v01.pdf (visited on 01/25/2022).

[Fidler 2003] Fidler, Markus (2003). "Extending the Network Calculus Pay Bursts Only Once Principle to Aggregate Scheduling." In: *Quality of Service in Multiservice IP Networks.* Ed. by Marco Ajmone Marsan, Giorgio Corazza, Marco Listanti, and Aldo Roveri. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, pp. 19–34. ISBN: 978-3-540-36480-1. DOI: 10.1007/3-540-36480-3_2.

[Fidler, Einhoff 2004] Fidler, Markus and Gerrit Einhoff (2004). "Routing in Turn-Prohibition Based Feed-Forward Networks." In: *Networking 2004.* Ed. by Nikolas Mitrou, Kimon Kontovasilis, George N. Rouskas, Ilias Iliadis, and Lazaros Merakos. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, pp. 1168–1179. ISBN: 978-3-540-24693-0. DOI: 10.1007/978-3-540-24693-0_96.

[Finn, Boudec, Mohammadpour, *et al.* 2022] Finn, Norman, Jean-Yves Le Boudec, Ehsan Mohammadpour, Jiayi Zhang, and Balazs Varga (Apr. 8, 2022). *DetNet Bounded Latency.* Internet Draft draft-ietf-detnet-bounded-latency-10. Work in Progress. Internet Engineering Task Force. 31 pp. URL: https://datatracker.ietf.org/doc/draft-ietf-detnet-bounded-latency (visited on 05/01/2022).

[Finzi, Craciunas 2019] Finzi, Anaïs and Silviu S. Craciunas (Nov. 6, 2019). "Breaking vs. Solving: Analysis and Routing of Real-Time Networks with Cyclic Dependencies Using Network Calculus." In: *Proceedings of the 27th International Conference on Real-Time Networks and Systems.* RTNS '19. New York, NY, USA: Association for Computing Machinery, pp. 101–111. ISBN: 978-1-4503-7223-7. DOI: 10.1145/3356401.3356418. URL: https://doi.org/10.1145/3356401.3356418 (visited on 11/24/2021).

[Freris, Graham, Kumar 2011] Freris, Nikolaos M., Scott R. Graham, and P. R. Kumar (June 2011). "Fundamental Limits on Synchronizing Clocks Over Networks." In: *IEEE Transactions on Automatic Control* 56.6, pp. 1352–1364. ISSN: 1558-2523. DOI: 10.1109/TAC.2010.2089210.

[Gaderer, Nagy, Loschmidt, *et al.* 2011] Gaderer, Georg, Anetta Nagy, Patrick Loschmidt, and Thilo Sauter (Jan. 1, 2011). "Achieving a Realistic Notion of Time in Discrete Event Simulation." In: *International Journal of Distributed Sensor Networks* 7.1, p. 294852. ISSN: 1550-1329. DOI: 10.1155/2011/294852. URL: https://doi.org/10.1155/2011/294852 (visited on 04/11/2022).

[Gamarnik 1998] Gamarnik, David (Nov. 8, 1998). "Stability of Adversarial Queues via Fluid Models." In: *Proceedings of the 39th Annual Symposium on Foundations of Computer Science.* FOCS '98. USA: IEEE Computer Society, p. 60. ISBN: 978-0-8186-9172-0.

[Garey, Johnson 1990] Garey, Michael R. and David S. Johnson (1990). *Computers and Intractability; A Guide to the Theory of NP-Completeness.* New York, NY, USA: W. H. Freeman & Co. ISBN: 978-0-7167-1045-5.

[Geyer, Bondorf 2019] Geyer, F. and S. Bondorf (Apr. 2019). "DeepTMA: Predicting Effective Contention Models for Network Calculus Using Graph Neural Networks." In: *IEEE INFOCOM 2019 - IEEE Conference on Computer Communications.* IEEE INFOCOM 2019 - IEEE Conference on Computer Communications, pp. 1009–1017. DOI: 10.1109/INFOCOM.2019.8737496.

[Geyer, Scheffler, Bondorf 2022] Geyer, Fabien, Alexander Scheffler, and Steffen Bondorf (Feb. 7, 2022). "Network Calculus with Flow Prolongation – A Feedforward FIFO Analysis Enabled by ML." arXiv: 2202.03004 [cs]. URL: http://arxiv.org/abs/2202.03004 (visited on 05/10/2022).

[Gollan, Zdarsky, Martinovic, *et al.* 2008] Gollan, Nicos, Frank A. Zdarsky, Ivan Martinovic, and Jens B. Schmitt (Mar. 2008). "The DISCO Network Calculator." In: *14th GI/ITG Conference - Measurement, Modelling and Evalutation of Computer and Communication Systems.* 14th GI/ITG Conference - Measurement, Modelling and Evalutation of Computer and Communication Systems, pp. 1–3.

[Grieu 2004] Grieu, Jérôme (Sept. 24, 2004). "Analyse et évaluation de techniques de commutation Ethernet pour l'interconnexion des systèmes avioniques." PhD thesis. URL: http://ethesis.inp-toulouse.fr/archive/00000084/ (visited on 07/22/2021).

[Hajek 2000] Hajek, B. (Jan. 2000). “Large Bursts Do Not Cause Instability.” In: *IEEE Transactions on Automatic Control* 45.1, pp. 116–118. ISSN: 1558-2523. DOI: 10.1109/9.827366.

[Heise 2018] Heise, Peter (2018). “Real-Time Guarantees, Dependability and Self-Configuration in Future Avionic Networks.” Siegen: Universitätsbibliothek der Universität Siegen.

[Heise, Geyer, Obermaisser 2016] Heise, Peter, Fabien Geyer, and Roman Obermaisser (Nov. 2016). “TSimNet: An Industrial Time Sensitive Networking Simulation Framework Based on OMNeT++.” In: *2016 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*. 2016 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS), pp. 1–5. DOI: 10.1109/NTMS.2016.7792488.

[Heise, Tobeck, Hanka, *et al.* 2014] Heise, Peter, Nils Tobeck, Oliver Hanka, and Stefan Schneele (Oct. 2014). “SAFDX: Deterministic High-Availability Ring for Industrial Low-Cost Networks.” In: *2014 7th International Workshop on Communication Technologies for Vehicles (Nets4Cars-Fall)*. 2014 7th International Workshop on Communication Technologies for Vehicles (Nets4Cars-Fall), pp. 40–44. DOI: 10.1109/Nets4CarsFall.2014.7000910.

[Hofmann, Nikolić, Ernst 2020] Hofmann, Robin, Borislav Nikolić, and Rolf Ernst (Dec. 2020). “Challenges and Limitations of IEEE 802.1CB-2017.” In: *IEEE Embedded Systems Letters* 12.4, pp. 105–108. ISSN: 1943-0671. DOI: 10.1109/LES.2019.2960744. URL: https://doi.org/10.1109/LES.2019.2960744.

[Karp 1972] Karp, Richard M. (1972). “Reducibility among Combinatorial Problems.” In: *Complexity of Computer Computations: Proceedings of a Symposium on the Complexity of Computer Computations, Held March 20–22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, and Sponsored by the Office of Naval Research, Mathematics Program, IBM World Trade Corporation, and the IBM Research Mathematical Sciences Department*. Ed. by Raymond E. Miller, James W. Thatcher, and Jean D. Bohlinger. The IBM Research Symposia Series. Boston, MA: Springer US, pp. 85–103. ISBN: 978-1-4684-2001-2. DOI: 10.1007/978-1-4684-2001-2_9. URL: https://doi.org/10.1007/978-1-4684-2001-2_9 (visited on 01/19/2022).

[Katevenis, Sidiropoulos, Courcoubetis 1991] Katevenis, M., S. Sidiropoulos, and C. Courcoubetis (Oct. 1991). “Weighted Round-Robin Cell Multiplexing in a General-Purpose ATM Switch Chip.” In: *IEEE Journal on Selected Areas in Communications* 9.8, pp. 1265–1279. ISSN: 1558-0008. DOI: 10.1109/49.105173.

[Kirrmann, Weber, Kleineberg, *et al.* 2009] Kirrmann, Hubert, Karl Weber, Oliver Kleineberg, and Hans Weibel (Sept. 2009). “HSR: Zero Recovery Time and Low-Cost Redundancy for Industrial Ethernet (High Availability Seamless Redundancy, IEC 62439-3).” In: *2009 IEEE Conference on Emerging Technologies Factory Automation*. 2009 IEEE Conference on Emerging Technologies Factory Automation, pp. 1–4. DOI: 10.1109/ETFA.2009.5347037.

[Krakora, Waszniowski, Pisa, *et al.* 2004] Krakora, J., L. Waszniowski, P. Pisa, and Z. Hanzalek (Sept. 2004). “Timed Automata Approach to Real Time Distributed System Verification.” In: *IEEE International Workshop on Factory Communication Systems, 2004. Proceedings.* IEEE International Workshop on Factory Communication Systems, 2004. Proceedings. Pp. 407–410. DOI: 10.1109/WFCS.2004.1377759.

[Le Boudec 1996] Le Boudec, Jean-Yves (1996). *Network Calculus Made Easy.* ECOLE POLYTECHNIQUE FEDERALE, LAUSANNE (EPFL.

[Le Boudec 2001] Le Boudec, Jean-Yves (June 1, 2001). “Some Properties of Variable Length Packet Shapers.” In: *ACM SIGMETRICS Performance Evaluation Review* 29.1, pp. 175–183. ISSN: 0163-5999. DOI: 10.1145/384268.378780. URL: https://doi.org/10.1145/384268.378780 (visited on 05/06/2022).

[Le Boudec 2018] – (Dec. 2018). “A Theory of Traffic Regulators for Deterministic Networks With Application to Interleaved Regulators.” In: *IEEE/ACM Transactions on Networking* 26.6, pp. 2721–2733. ISSN: 1063-6692. DOI: 10.1109/TNET.2018.2875191. URL: http://doi.org/10.1109/TNET.2018.2875191.

[Le Boudec, Thiran 2001] Le Boudec, Jean-Yves and Patrick Thiran (2001). *Network Calculus: A Theory of Deterministic Queuing Systems for the Internet.* Lecture Notes in Computer Science, Lect.Notes Computer. Tutorial. Berlin Heidelberg: Springer-Verlag. ISBN: 978-3-540-42184-9. URL: https://www.springer.com/us/book/9783540421849 (visited on 02/04/2019).

[Lenzini, Martorini, Mingozzi, *et al.* 2006] Lenzini, Luciano, Linda Martorini, Enzo Mingozzi, and Giovanni Stea (Oct. 1, 2006). “Tight End-to-End per-Flow Delay Bounds in FIFO Multiplexing Sink-Tree Networks.” In: *Performance Evaluation* 63.9, pp. 956–987. ISSN: 0166-5316. DOI: 10.1016/j.peva.2005.10.003. URL: https://www.sciencedirect.com/science/article/pii/S0166531605001537 (visited on 05/02/2022).

[Lenzini, Mingozzi, Stea 2007] Lenzini, Luciano, Enzo Mingozzi, and Giovanni Stea (Oct. 22, 2007). “End-to-End Delay Bounds in FIFO-multiplexing Tandems.” In: *Proceedings of the 2nd International Conference on Performance Evaluation Methodologies and Tools.* ValueTools ’07. Brussels, BEL: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), pp. 1–10. ISBN: 978-963-9799-00-4.

[Lenzini, Mingozzi, Stea 2008] – (Nov. 1, 2008). “A Methodology for Computing End-to-End Delay Bounds in FIFO-multiplexing Tandems.” In: *Performance Evaluation.* Performance Evaluation Methodologies and Tools: Selected Papers from ValueTools 2007 65.11, pp. 922–943. ISSN: 0166-5316. DOI: 10.1016/j.peva.2008.04.002. URL: https://www.sciencedirect.com/science/article/pii/S016653160800028X (visited on 05/09/2022).

[Li, Zhu, Savaria, *et al.* 2017] Li, Meng, Guchuan Zhu, Yvon Savaria, and Michaël Lauer (Oct. 2017). “Reliability Enhancement of Redundancy Management in AFDX Networks.” In: *IEEE Transactions on Industrial Informatics* 13.5, pp. 2118–2129. ISSN: 1941-0050. DOI: 10.1109/TII.2017.2732345.

[Li, Cros, George 2014] Li, Xiaoting, Olivier Cros, and Laurent George (Aug. 2014). “The Trajectory Approach for AFDX FIFO Networks Revisited and Corrected.” In: *The 20th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications.* Chongqing, China: IEEE. URL: https://hal-upec-upem.archives-ouvertes.fr/hal-00975730 (visited on 06/27/2022).

[Li, Gu 2009] Li, Y. and H. Gu (Oct. 2009). “XY-turn Model for Deadlock Free Routing in Honeycomb Networks-on-Chip.” In: *2009 15th Asia-Pacific Conference on Communications.* 2009 15th Asia-Pacific Conference on Communications, pp. 900–903. DOI: 10.1109/APCC.2009.5375521. URL: http://doi.org/10.1109/APCC.2009.5375521.

[Liebeherr 2017] Liebeherr, Jörg (2017). *Duality of the Max-Plus and Min-Plus Network Calculus.* now. URL: https://ieeexplore.ieee.org/document/8187214 (visited on 11/08/2019).

[Liu, Layland 1973] Liu, C. L. and James W. Layland (Jan. 1, 1973). “Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment.” In: *Journal of the ACM* 20.1, pp. 46–61. ISSN: 0004-5411. DOI: 10.1145/321738.321743. URL: https://doi.org/10.1145/321738.321743 (visited on 05/29/2022).

[Mahmood, Exel, Trsek, *et al.* 2017] Mahmood, Aneeq, Reinhard Exel, Henning Trsek, and Thilo Sauter (Apr. 2017). “Clock Synchronization Over IEEE 802.11—A Survey of Methodologies and Protocols.” In: *IEEE Transactions on Industrial Informatics* 13.2, pp. 907–922. ISSN: 1941-0050. DOI: 10.1109/TII.2016.2629669.

[Maile, Hielscher, German 2020] Maile, Lisa, Kai-Steffen Hielscher, and Reinhard German (May 2020). “Network Calculus Results for TSN: An Introduction.” In: *2020 Information Communication Technologies Conference (ICTC)*. 2020 Information Communication Technologies Conference (ICTC). 41, 50 ,29,26,28, pp. 131–140. DOI: 10.1109/ICTC49638.2020.9123308.

[Martin 2004] Martin, Steven (July 2004). “Maîtrise de La Dimension Temporelle de La Qualité de Service Dans Les Réseaux.” Francis COTTET (ENSMA)¡br /¿Françoise SIMONOT-LION (ENSMN)¡br /¿Yacine AMIRAT(Université Paris 12)¡br /¿Laurent GEORGE (Université Paris 12)¡br /¿Pascal LORENZ (Université de Haute Alsace)¡br /¿Pascale MINET (INRIA Rocquencourt¡br /¿Samir TOHME (Université de Versailles St-Quentin). Theses. Université Paris XII Val de Marne. URL: https://tel.archives-ouvertes.fr/tel-00007638.

[Maruyama, Yamada, Yoshida, *et al.* 2015] Maruyama, Tatsuya, Tsutomu Yamada, Shouji Yoshida, Mitsuyasu Kido, and Chikashi Komatsu (Oct. 2015). “NS-3 Based IEEE 1588 Synchronization Simulator for Multi-Hop Network.” In: *2015 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS)*. 2015 IEEE International Symposium on Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS), pp. 99–104. DOI: 10.1109/ISPCS.2015.7324691.

[Mifdaoui, Ayed 2010] Mifdaoui, Ahlem and Hamdi Ayed (Dec. 2010). “WOPANets: A Tool for WOrst Case Performance Analysis of Embedded Networks.” In: *2010 15th IEEE International Workshop on Computer Aided Modeling, Analysis and Design of Communication Links and Networks (CAMAD)*. 2010 15th IEEE International Workshop on Computer Aided Modeling, Analysis and Design of Communication Links and Networks (CAMAD), pp. 91–95. DOI: 10.1109/CAMAD.2010.5686958.

[Mifdaoui, Leydier 2017] Mifdaoui, Ahlem and Thierry Leydier (Dec. 2017). “Beyond the Accuracy-Complexity Tradeoffs of Compositional Analyses Using Network Calculus for Complex Networks.” In: *10th International Workshop on Compositional Theory and Technology for Real-Time Embedded Systems (Co-Located with RTSS 2017)*. Paris, France, pp. 1–8. URL: https://hal.archives-ouvertes.fr/hal-01690096 (visited on 04/12/2019).

[Mohammadpour, Stai, Boudec 2019] Mohammadpour, E., E. Stai, and J. Le Boudec (Sept. 2019). “Improved Credit Bounds for the Credit-Based Shaper in Time-Sensitive Networking.” In: *IEEE Networking Letters* 1.3, pp. 136–139. DOI: 10.1109/LNET.2019.2925176. URL: http://doi.org/10.1109/LNET.2019.2925176.

[Mohammadpour, Stai, Le Boudec 2019] Mohammadpour, E., E. Stai, and J.-Y. Le Boudec (2019). “Improved Delay Bound for a Service Curve Element with Known Transmission Rate.”

In: *IEEE Networking Letters*, pp. 1–1. DOI: 10.1109/LNET.2019.2927143. URL: http://doi.org/10.1109/LNET.2019.2927143.

[Mohammadpour, Stai, Mohiuddin, *et al.* 2018] Mohammadpour, E., E. Stai, M. Mohiuddin, and J.-Y. Le Boudec (Sept. 2018). "Latency and Backlog Bounds in Time-Sensitive Networking with Credit Based Shapers and Asynchronous Traffic Shaping." In: *2018 30th International Teletraffic Congress (ITC 30)*. 2018 30th International Teletraffic Congress (ITC 30). Vol. 02, pp. 1–6. DOI: 10.1109/ITC30.2018.10053. URL: http://doi.org/10.1109/ITC30.2018.10053.

[Mohammadpour, Le Boudec 2021] Mohammadpour, Ehsan and Jean-Yves Le Boudec (2021). "On Packet Reordering in Time-Sensitive Networks." In: *IEEE/ACM Transactions on Networking*, pp. 1–13. ISSN: 1558-2566. DOI: 10.1109/TNET.2021.3129590.

[Moreira, Serrano, Wlostowski, *et al.* 2009] Moreira, P., J. Serrano, T. Wlostowski, P. Loschmidt, and G. Gaderer (Oct. 2009). "White Rabbit: Sub-nanosecond Timing Distribution over Ethernet." In: *Control and Communication 2009 International Symposium on Precision Clock Synchronization for Measurement*. Control and Communication 2009 International Symposium on Precision Clock Synchronization for Measurement, pp. 1–5. DOI: 10.1109/ISPCS.2009.5340196.

[Murta, Torres Jr. Mohapatra 2006] Murta, Cristina D., Pedro R. Torres Jr., and Prasant Mohapatra (Nov. 2006). "QRPp1-4: Characterizing Quality of Time and Topology in a Time Synchronization Network." In: *IEEE Globecom 2006*. IEEE Globecom 2006, pp. 1–5. DOI: 10.1109/GLOCOM.2006.467.

[Nasrallah, Thyagaturu, Alharbi, *et al.* 2019] Nasrallah, Ahmed, Akhilesh S. Thyagaturu, Ziyad Alharbi, Cuixiang Wang, Xing Shao, Martin Reisslein, and Hesham Elbakoury (2019). "Performance Comparison of IEEE 802.1 TSN Time Aware Shaper (TAS) and Asynchronous Traffic Shaper (ATS)." In: *IEEE Access* 7, pp. 44165–44181. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2908613.

[Navet, Song, Simonot-Lion, *et al.* 2005] Navet, N., Y. Song, F. Simonot-Lion, and C. Wilwert (June 2005). "Trends in Automotive Communication Systems." In: *Proceedings of the IEEE* 93.6, pp. 1204–1223. ISSN: 1558-2256. DOI: 10.1109/JPROC.2005.849725.

[Navet, Bengtsson, Migge 2020] Navet, Nicolas, Hoai Hoang Bengtsson, and Jörn Migge (Feb. 12, 2020). "Early-Stage Bottleneck Identification and Removal in TSN Networks." Automotive Ethernet Congress. URL: https://orbilu.uni.lu/handle/10993/46282 (visited on 04/13/2021).

[NAVET, MAI, MIGGE 2019] NAVET, Nicolas, Tieu Long MAI, and Jörn MIGGE (Jan. 29, 2019). *Using Machine Learningto SpeedUp the Design Space Exploration of Ethernet TSN Networks*. RTaW. URL: https://orbilu.uni.lu/bitstream/10993/38604/1/feasibility-with-ml.pdf (visited on 04/06/2020).

[Navet, Simonot-Lion 2013] Navet, Nicolas and Françoise Simonot-Lion (2013). "In-Vehicle Communication Networks - a Historical Perspective and Review." In: *Industrial Communication Technology Handbook, Second Edition*. Ed. by Richard Zurawski. CRC Press Taylor&Francis. URL: https://hal.inria.fr/hal-00876524.

[Pahlevan, Obermaisser 2018] Pahlevan, Maryam and Roman Obermaisser (Nov. 2018). "Redundancy Management for Safety-Critical Applications with Time Sensitive Networking." In: *2018 28th International Telecommunication Networks and Applications Conference (ITNAC)*.

2018 28th International Telecommunication Networks and Applications Conference (ITNAC), pp. 1–7. DOI: 10.1109/ATNAC.2018.8615374.

[Plassart, Le Boudec 2021] Plassart, Stéphan and Jean-Yves Le Boudec (Nov. 2, 2021). "Equivalent Versions of Total Flow Analysis." arXiv: 2111.01827 [cs]. URL: http://arxiv.org/abs/2111.01827 (visited on 01/09/2022).

[Pop, Eles, Peng 1999] Pop, P., P. Eles, and Z. Peng (Mar. 1999). "Scheduling with Optimized Communication for Time-Triggered Embedded Systems." In: *Proceedings of the Seventh International Workshop on Hardware/Software Codesign (CODES'99) (IEEE Cat. No.99TH8450).* Proceedings of the Seventh International Workshop on Hardware/Software Codesign (CODES'99) (IEEE Cat. No.99TH8450), pp. 178–182. DOI: 10.1145/301177.303812.

[Pop, Eles, Peng 2003] Pop, P., P. Eles, and Zebo Peng (Mar. 2003). "Schedulability Analysis and Optimization for the Synthesis of Multi-Cluster Distributed Embedded Systems." In: *Automation and Test in Europe Conference and Exhibition 2003 Design.* Automation and Test in Europe Conference and Exhibition 2003 Design, pp. 184–189. DOI: 10.1109/DATE.2003.1253606.

[Pop, Eles, Peng 2004] Pop, Paul, Petru Eles, and Zebo Peng (Apr. 1, 2004). "Schedulability-Driven Communication Synthesis for Time Triggered Embedded Systems." In: *Real-Time Systems* 26.3, pp. 297–325. ISSN: 1573-1383. DOI: 10.1023/B:TIME.0000018246.09796.a3. URL: https://doi.org/10.1023/B:TIME.0000018246.09796.a3 (visited on 05/29/2022).

[Pop, Raagaard, Craciunas, *et al.* 2016] Pop, Paul, Michael Lander Raagaard, Silviu S. Craciunas, and Wilfried Steiner (2016). "Design Optimisation of Cyber-Physical Distributed Systems Using IEEE Time-Sensitive Networks." In: *IET Cyber-Physical Systems: Theory & Applications* 1.1, pp. 86–94. ISSN: 2398-3396. DOI: 10.1049/iet-cps.2016.0021. URL: https://onlinelibrary.wiley.com/doi/abs/10.1049/iet-cps.2016.0021 (visited on 05/29/2022).

[Pop, Pop, Eles, *et al.* 2005] Pop, T., P. Pop, P. Eles, and Zebo Peng (Aug. 2005). "Optimization of Hierarchically Scheduled Heterogeneous Embedded Systems." In: *11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'05).* 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA'05), pp. 67–71. DOI: 10.1109/RTCSA.2005.67.

[Powers, Hahn 2004] Powers, Edward and Jorg Hahn (Apr. 2004). "GPS and Galileo UTC Time Distribution." In: *2004 18th European Frequency and Time Forum (EFTF 2004).* 2004 18th European Frequency and Time Forum (EFTF 2004), pp. 484–488. DOI: 10.1049/cp:20040914.

[Raagaard, Pop 2017] Raagaard, Michael Lander and Paul Pop (Jan. 2017). *Optimization Algorithms for the Scheduling of IEEE 802.1 Time-Sensitive Networking (TSN).* 2800 Kongens Lyngby, Denmar: Technical University of Denmark. URL: http://www2.compute.dtu.dk/~paupo/publications/Raagaard2017aa-Optimization%20algorithms%20for%20th-.pdf (visited on 05/26/2022).

[Renczes, Kovácsházy 2020] Renczes, Balázs and Tamás Kovácsházy (May 2020). "An Enhanced Oscillator Model for Clock Synchronization Protocols." In: *2020 IEEE International Instrumentation and Measurement Technology Conference (I2MTC).* 2020 IEEE International Instrumentation and Measurement Technology Conference (I2MTC), pp. 1–5. DOI: 10.1109/I2MTC43012.2020.9128710.

[Ridoux, Veitch 2010] Ridoux, Julien and Darryl Veitch (May 1, 2010). “Principles of Robust Timing over the Internet.” In: *Communications of the ACM*. URL: https://dl.acm.org/doi/abs/10.1145/1735223.1735241 (visited on 04/06/2020).

[Rizzo, Le Boudec 2008] Rizzo, G. and J.-Y. Le Boudec (Apr. 2008). “Stability and Delay Bounds in Heterogeneous Networks of Aggregate Schedulers.” In: *IEEE INFOCOM 2008 - The 27th Conference on Computer Communications*. IEEE INFOCOM 2008 - The 27th Conference on Computer Communications, pp. 1490–1498. DOI: 10.1109/INFOCOM.2008.208.

[Rizzo, Le Boudec 2007] Rizzo, Gianluca and Jean-Yves Le Boudec (Nov. 2007). “Generalization of the RIN Result to Heterogeneous Networks of Aggregate Schedulers and Leaky Bucket Constrained Flows.” In: *2007 15th IEEE International Conference on Networks*. 2007 15th IEEE International Conference on Networks, pp. 388–393. DOI: 10.1109/ICON.2007.4444118.

[Roedig, Gollan, Schmitt 2007] Roedig, Utz, Nicos Gollan, and Jens Schmitt (Oct. 22, 2007). “Validating the Sensor Network Calculus by Simulations.” In: p. 34. ISSN: 978-963-9799-12-7. DOI: 10.4108/pwsn.2007.2290.

[Samson, Vergnaud, Dujardin, *et al.* 2021] Samson, Maxime, Thomas Vergnaud, Éric Dujardin, Laurent Ciarletta, and Ye-Qiong Song (Sept. 2021). “Une Approche de Génération Automatique de Configuration Basée Sur Les Modèles Pour Les Réseaux TSN.” In: *#ETR2021 - L’École d’Été Temps Réel 2021*. Poitiers, France: LIAS-ISAE/ENSMA. URL: https://hal.archives-ouvertes.fr/hal-03573608 (visited on 05/20/2022).

[Schmitt, Zdarsky, Fidler 2008] Schmitt, J. B., F. A. Zdarsky, and M. Fidler (Apr. 2008). “Delay Bounds under Arbitrary Multiplexing: When Network Calculus Leaves You in the Lurch...” In: *IEEE INFOCOM 2008 - The 27th Conference on Computer Communications*. IEEE INFOCOM 2008 - The 27th Conference on Computer Communications, pp. 1669–1677. DOI: 10.1109/INFOCOM.2008.228.

[Schmitt, Zdarsky, Roedig 2006] Schmitt, Jens, Frank Zdarsky, and Utz Roedig (Jan. 1, 2006). “Sensor Network Calculus with Multiple Sinks.” In: *Proceedings of IFIP Networking Workshop on Performance Control in Wireless Sensor Networks (PWSN’06)*.

[Schmitt, Roedig 2005] Schmitt, Jens B. and Utz Roedig (2005). “Sensor Network Calculus – A Framework for Worst Case Analysis.” In: *Distributed Computing in Sensor Systems*. Ed. by Viktor K. Prasanna, Sitharama S. Iyengar, Paul G. Spirakis, and Matt Welsh. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, pp. 141–154. ISBN: 978-3-540-31671-8. DOI: 10.1007/11502593_13.

[Schmitt, Zdarsky 2006] Schmitt, Jens B. and Frank A. Zdarsky (Oct. 11, 2006). “The DISCO Network Calculator: A Toolbox for Worst Case Analysis.” In: *Proceedings of the 1st International Conference on Performance Evaluation Methodolgies and Tools*. Valuetools ’06. New York, NY, USA: Association for Computing Machinery, 8–es. ISBN: 978-1-59593-504-5. DOI: 10.1145/1190095.1190105. URL: https://doi.org/10.1145/1190095.1190105 (visited on 09/25/2020).

[Schmitt, Zdarsky, Martinovic 2008] Schmitt, Jens B., Frank A. Zdarsky, and Ivan Martinovic (Mar. 2008). “Improving Performance Bounds in Feed-Forward Networks by Paying Multiplexing Only Once.” In: *14th GI/ITG Conference - Measurement, Modelling and Evalutation of Computer and Communication Systems*. 14th GI/ITG Conference - Measurement, Modelling and Evalutation of Computer and Communication Systems, pp. 1–15.

[Shreedhar, Varghese 1995] Shreedhar, M. and George Varghese (Oct. 1, 1995). “Efficient Fair Queueing Using Deficit Round Robin.” In: *ACM SIGCOMM Computer Communication Review* 25.4, pp. 231–242. ISSN: 0146-4833. DOI: 10.1145/217391.217453. URL: https://doi.org/10.1145/217391.217453 (visited on 04/28/2022).

[Song, Koubaa, Simonot 2002] Song, Yeqiong, A. Koubaa, and F. Simonot (Aug. 2002). “Switched Ethernet for Real-Time Industrial Communication: Modelling and Message Buffering Delay Evaluation.” In: *4th IEEE International Workshop on Factory Communication Systems*. 4th IEEE International Workshop on Factory Communication Systems, pp. 27–35. DOI: 10.1109/WFCS.2002.1159697.

[Soni, Li, Scharbarg, *et al.* 2018] Soni, Aakash, Xiaoting Li, Jean-Luc Scharbarg, and Christian Fraboul (Oct. 10, 2018). “WCTT Analysis of Avionics Switched Ethernet Network with WRR Scheduling.” In: *Proceedings of the 26th International Conference on Real-Time Networks and Systems*. RTNS ’18. New York, NY, USA: Association for Computing Machinery, pp. 213–222. ISBN: 978-1-4503-6463-8. DOI: 10.1145/3273905.3273925. URL: https://doi.org/10.1145/3273905.3273925 (visited on 04/28/2022).

[Specht, Samii 2016] Specht, Johannes and Soheil Samii (July 2016). “Urgency-Based Scheduler for Time-Sensitive Switched Ethernet Networks.” In: *2016 28th Euromicro Conference on Real-Time Systems (ECRTS)*. 2016 28th Euromicro Conference on Real-Time Systems (ECRTS), pp. 75–85. DOI: 10.1109/ECRTS.2016.27.

[Starobinski, Karpovsky, Zakrevski 2003] Starobinski, D., M. Karpovsky, and L. A. Zakrevski (June 2003). “Application of Network Calculus to General Topologies Using Turn-Prohibition.” In: *IEEE/ACM Transactions on Networking* 11.3, pp. 411–421. ISSN: 1063-6692. DOI: 10.1109/TNET.2003.813040. URL: http://doi.org/10.1109/TNET.2003.813040.

[Tabatabaee, Le Boudec 2022] Tabatabaee, Seyed Mohammadhossein and Jean-Yves Le Boudec (2022). “Deficit Round-Robin: A Second Network Calculus Analysis.” In: *IEEE/ACM Transactions on Networking*, pp. 1–15. ISSN: 1558-2566. DOI: 10.1109/TNET.2022.3164772.

[Tabatabaee, Le Boudec, Boyer 2021] Tabatabaee, Seyed Mohammadhossein, Jean-Yves Le Boudec, and Marc Boyer (2021). “Interleaved Weighted Round-Robin: A Network Calculus Analysis.” In: *IEICE Transactions on Communications* E104.B.12, pp. 1479–1493. DOI: 10.1587/transcom.2021ITI0001.

[Tamas-Selicean, Pop, Steiner 2012] Tamas-Selicean, Domitian, Paul Pop, and Wilfried Steiner (Oct. 7, 2012). “Synthesis of Communication Schedules for TTEthernet-based Mixed-Criticality Systems.” In: *Proceedings of the Eighth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis*. CODES+ISSS ’12. New York, NY, USA: Association for Computing Machinery, pp. 473–482. ISBN: 978-1-4503-1426-8. DOI: 10.1145/2380445.2380518. URL: https://doi.org/10.1145/2380445.2380518 (visited on 05/29/2022).

[Tămaş–Selicean, Pop, Steiner 2015] Tămaş–Selicean, Domiţian, Paul Pop, and Wilfried Steiner (Jan. 1, 2015). “Design Optimization of TTEthernet-based Distributed Real-Time Systems.” In: *Real-Time Systems* 51.1, pp. 1–35. ISSN: 1573-1383. DOI: 10.1007/s11241-014-9214-8. URL: https://doi.org/10.1007/s11241-014-9214-8 (visited on 05/29/2022).

[Tassiulas, Georgiadis 1994] Tassiulas, L. and L. Georgiadis (June 1994). “Any Work-Conserving Policy Stabilizes the Ring with Spatial Reuse.” In: *Proceedings of INFOCOM ’94 Conference* *on Computer Communications*. Proceedings of INFOCOM '94 Conference on Computer Communications, 66–70 vol.1. DOI: 10.1109/INFCOM.1994.337631.

[Thomas, Le Boudec 2020] Thomas, Ludovic and Jean-Yves Le Boudec (June 9, 2020). "On Time Synchronization Issues in Time-Sensitive Networks with Regulators and Nonideal Clocks." In: *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 4.2, 27:1–27:41. DOI: 10.1145/3392145.

[Thomas, Le Boudec, Mifdaoui 2019] Thomas, Ludovic, Jean-Yves Le Boudec, and Ahlem Mifdaoui (Dec. 2019). "On Cyclic Dependencies and Regulators in Time-Sensitive Networks." In: *2019 IEEE Real-Time Systems Symposium (RTSS)*. 2019 IEEE Real-Time Systems Symposium (RTSS), pp. 299–311. DOI: 10.1109/RTSS46320.2019.00035.

[Thomas, Mifdaoui, Le Boudec 2022] Thomas, Ludovic, Ahlem Mifdaoui, and Jean-Yves Le Boudec (2022). "Worst-Case Delay Bounds in Time-Sensitive Networks With Packet Replication and Elimination." In: *IEEE/ACM Transactions on Networking*, pp. 1–15. ISSN: 1558-2566. DOI: 10.1109/TNET.2022.3180763.

[Tindell, Hansson, Wellings 1994] Tindell, Hansson, and Wellings (Dec. 1994). "Analysing Real-Time Communications: Controller Area Network (CAN)." In: *1994 Proceedings Real-Time Systems Symposium*. 1994 Proceedings Real-Time Systems Symposium, pp. 259–263. DOI: 10.1109/REAL.1994.342710.

[Tindell, Clark 1994] Tindell, Ken and John Clark (Apr. 1, 1994). "Holistic Schedulability Analysis for Distributed Hard Real-Time Systems." In: *Microprocessing and Microprogramming*. Parallel Processing in Embedded Real-time Systems 40.2, pp. 117–134. ISSN: 0165-6074. DOI: 10.1016/0165-6074(94)90080-9. URL: https://www.sciencedirect.com/science/article/pii/0165607494900809 (visited on 05/29/2022).

[Varga, Farkas, Kehrer, *et al.* 2021] Varga, Balazs, János Farkas, Stephan Kehrer, and Tobias Heer (May 21, 2021). *Deterministic Networking (DetNet): Packet Ordering Function*. Internet-draft draft-varga-detnet-pof-01. Work in Progress. Internet Engineering Task Force / Internet Engineering Task Force. 11 pp. URL: https://datatracker.ietf.org/doc/html/draft-varga-detnet-pof-01.

[Wagner 2001] Wagner, Kurt (2001). "Short Evaluation of Linux's Token-Bucket-Filter (TBF) Queuing Discipline." In: *http://www.docum.org/stef.coene/qos/docs/other/tbf02_kw.ps*.

[Wandeler, Maxiaguine, Thiele 2006] Wandeler, E., A. Maxiaguine, and L. Thiele (Mar. 2006). "Performance Analysis of Greedy Shapers in Real-Time Systems." In: *Proceedings of the Design Automation Test in Europe Conference*. Proceedings of the Design Automation Test in Europe Conference. Vol. 1, 6 pp.-. DOI: 10.1109/DATE.2006.243801. URL: http://doi.org/10.1109/DATE.2006.243801.

[Wilwert, Navet, Song, *et al.* 2005] Wilwert, Cédric, Nicolas Navet, Ye-Qiong Song, and Françoise Simonot-Lion (2005). "Design of Automotive X-by-Wire Systems." In: *The Industrial Communication Technology Handbook*. Ed. by Richard Zurawski. http://www.taylorandfrancis.com/. CRC Press. URL: https://hal.inria.fr/inria-00000562 (visited on 05/29/2022).

[Witsch, Vogel-Heuser, Faure, *et al.* 2006] Witsch, Daniel, Birgit Vogel-Heuser, Jean-Marc Faure, and Gaëlle Marsal (Jan. 1, 2006). "PERFORMANCE ANALYSIS OF INDUSTRIAL ETHERNET NETWORKS BY MEANS OF TIMED MODEL-CHECKING." In: *IFAC Proceedings Volumes*. 12th IFAC Symposium on Information Control Problems in Manufacturing 39.3, pp. 101–106. ISSN: 1474-6670. DOI: 10.3182/20060517-3-FR-2903.00063. URL:

https://www.sciencedirect.com/science/article/pii/S1474667015357839 (visited on 05/29/2022).

[Zhao, Pop, Zheng, *et al.* 2018] Zhao, L., P. Pop, Z. Zheng, and Q. Li (Apr. 2018). "Timing Analysis of AVB Traffic in TSN Networks Using Network Calculus." In: *2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. 2018 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS), pp. 25–36. DOI: 10.1109/RTAS.2018.00009. URL: http://doi.org/10.1109/RTAS.2018.00009.

[Zhao, Pop, Craciunas 2018] Zhao, Luxi, Paul Pop, and Silviu S. Craciunas (2018). "Worst-Case Latency Analysis for IEEE 802.1Qbv Time Sensitive Networks Using Network Calculus." In: *IEEE Access* 6, pp. 41803–41815. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2018.2858767.

[Zhao, Pop, Steinhorst 2021] Zhao, Luxi, Paul Pop, and Sebastian Steinhorst (Mar. 24, 2021). "Quantitative Performance Comparison of Various Traffic Shapers in Time-Sensitive Networking." arXiv: 2103.13424 [cs]. URL: http://arxiv.org/abs/2103.13424 (visited on 06/29/2021).

[Zhao, Pop, Zheng, *et al.* 2021] Zhao, Luxi, Paul Pop, Zhong Zheng, Hugo Daigmorte, and Marc Boyer (Oct. 2021). "Latency Analysis of Multiple Classes of AVB Traffic in TSN With Standard Credit Behavior Using Network Calculus." In: *IEEE Transactions on Industrial Electronics* 68.10, pp. 10291–10302. ISSN: 1557-9948. DOI: 10.1109/TIE.2020.3021638.

[Zhu, Ma, Ryu 2013] Zhu, Hua, Liangping Ma, and Bong K. Ryu (Feb. 19, 2013). "System and Method for Clock Modeling in Discrete-Event Simulation." U.S. pat. 8380482B2. Boeing Co. URL: https://patents.google.com/patent/US8380482B2/en?oq=US+8380482+B2+ (visited on 04/11/2022).

[Zippo, Stea 2022] Zippo, Raffaele and Giovanni Stea (May 23, 2022). "Nancy: An Efficient Parallel Network Calculus Library." Comment: Preprint submitted to Software Impacts. DOI: 10.48550/arXiv.2205.11449. arXiv: 2205.11449 [cs]. URL: http://arxiv.org/abs/2205.11449 (visited on 06/07/2022).

[1] Aguirre Rodrigo, Guillermo and Ludovic Thomas (June 2020). *Clock: Module for Local Clock Implementation (!332) · Merge Request · Nsnam / Ns-3-Dev.* GitLab. URL: https://gitlab.com/nsnam/ns-3-dev/-/merge_requests/332 (visited on 04/10/2022).

[2] *Ansys SCADE Suite — Model-Based Design* (2022). URL: https://www.ansys.com/products/embedded-software/ansys-scade-suite (visited on 05/29/2022).

[3] Barnes, Peter D. (Fri Nov 12 16:29:31 PST 2021). *[Ns-developers] [MR #332] Local Clock Module.* E-mail. URL: https://mailman.isi.edu/pipermail/ns-developers/2021-November/015584.html (visited on 04/19/2022).

[RFC 2475] Black, David L., Zheng Wang, Mark A. Carlson, Walter Weiss, Elwyn B. Davies, and Steven L. Blake (Dec. 1998). *An Architecture for Differentiated Services.* Request for Comments RFC 2475. Internet Engineering Task Force. 36 pp. DOI: 10.17487/RFC2475. URL: https://datatracker.ietf.org/doc/rfc2475 (visited on 09/28/2021).

[4] *Clock Model — INET 4.3.0 Documentation* (2022). URL: https://inet.omnetpp.org/docs/users-guide/ch-clock.html (visited on 04/12/2022).

[AFDX] Committee, AEE et al. (2002). "Aircraft Data Network Part 7, Avionics Full Duplex Switched Ethernet (AFDX) Network, ARINC Specification 664." In: *Annapolis, Maryland: Aeronautical Radio.*

[5] *Core* (2022). *Core: SimulatorAdapter to Simplify Customizations on Top of SimulatorImpl (!807) · Merge Requests · Nsnam / Ns-3-Dev.* GitLab. URL: https://gitlab.com/nsnam/ns-3-dev/-/merge_requests/807 (visited on 04/19/2022).

[6] EPFL, LCA2 (Apr. 7, 2022). *Simulation of Instability in Time-Sensitive Networks with Regulators and Imperfect Clocks Using Ns-3.* URL: https://github.com/LCA2-EPFL/TSN-ATS-Clocks/blob/10c295ca2849c7828ff13f7828d651575bc3504a/src/clock/model/adversarial-clock.cc (visited on 04/19/2022).

[48] Finn, Norman, Jean-Yves Le Boudec, Ehsan Mohammadpour, Jiayi Zhang, and Balazs Varga (Apr. 8, 2022). *DetNet Bounded Latency.* Internet Draft draft-ietf-detnet-bounded-latency-10. Work in Progress. Internet Engineering Task Force. 31 pp. URL: https://datatracker.ietf.org/doc/draft-ietf-detnet-bounded-latency (visited on 05/01/2022).

[RFC 8655] Finn, Norman, Pascal Thubert, Balázs Varga, and János Farkas (2019). "Deterministic Networking Architecture." In: RFC 8655. ISSN: 2070-1721. DOI: 10.17487/RFC8655. URL: https://www.rfc-editor.org/info/rfc8655 (visited on 06/07/2021).

[7] Henderson, Tom (Tue Jun 30 07:09:32 PDT 2020). *[Ns-developers] [MR #332] Local Clock Module.* E-mail. URL: https://mailman.isi.edu/pipermail/ns-developers/2020-June/015222.html (visited on 04/19/2022).

[IEC 62439-3] *IEC 62439-3: Industrial Communication Networks : High Availability Automation Networks. Parallel Redundancy Protocol (PRP) and High Availability Seamless Redundancy (HSR).* (2010). ptie. 3. IEC. URL: https://books.google.ch/books?id=G3MujwEACAAJ.

[IEC/IEEE 60802] *IEC/IEEE 60802 TSN Profile for Industrial Automation —* (2022). URL: https://1.ieee802.org/tsn/iec-ieee-60802/ (visited on 01/01/2022).

[IEEE 1139] "IEEE Standard Definitions of Physical Quantities for Fundamental Frequency and Time Metrology–Random Instabilities" (Feb. 2009). In: *IEEE Std 1139-2008 (Revision of IEEE Std 1139-1999)*, pp. 1–50. DOI: 10.1109/IEEESTD.2009.6581834.

[IEEE 1588] "IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems" (July 2008). In: *IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002)*, pp. 1–269. DOI: 10.1109/IEEESTD.2008.4579760.

[IEEE 802.3] "IEEE Standard for Ethernet" (Aug. 2018). In: *IEEE Std 802.3-2018 (Revision of IEEE Std 802.3-2015)*, pp. 1–5600. DOI: 10.1109/IEEESTD.2018.8457469.

[IEEE 802.1Q] "IEEE Standard for Local and Metropolitan Area Network–Bridges and Bridged Networks" (July 2018). In: *IEEE Std 802.1Q-2018 (Revision of IEEE Std 802.1Q-2014)*, pp. 1–1993. DOI: 10.1109/IEEESTD.2018.8403927.

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

[IEEE 802.1CM] "IEEE Standard for Local and Metropolitan Area Networks – Time-Sensitive Networking for Fronthaul" (June 2018). In: *IEEE Std 802.1CM-2018*, pp. 1–62. DOI: 10.1109/IEEESTD.2018.8376066.

[IEEE 802.1Qav] "IEEE Standard for Local and Metropolitan Area Networks– Virtual Bridged Local Area Networks Amendment 12" (Jan. 2010). "IEEE Standard for Local and Metropolitan Area Networks– Virtual Bridged Local Area Networks Amendment 12: Forwarding and Queuing Enhancements for Time-Sensitive Streams." In: *IEEE Std 802.1Qav-2009 (Amendment to IEEE Std 802.1Q-2005)*, pp. 1–72. DOI: 10.1109/IEEESTD.2010.8684664.

[IEEE 802.1BA] "IEEE Standard for Local and Metropolitan Area Networks–Audio Video Bridging (AVB) Systems" (Sept. 2011). In: *IEEE Std 802.1BA-2011*, pp. 1–45. DOI: 10.1109/IEEESTD.2011.6032690.

[IEEE 802.1Qcr] "IEEE Standard for Local and Metropolitan Area Networks–Bridges and Bridged Networks - Amendment 34" (Nov. 2020). "IEEE Standard for Local and Metropolitan Area Networks–Bridges and Bridged Networks - Amendment 34:Asynchronous Traffic Shaping." In: *IEEE Std 802.1Qcr-2020 (Amendment to IEEE Std 802.1Q-2018 as amended by IEEE Std 802.1Qcp-2018, IEEE Std 802.1Qcc-2018, IEEE Std 802.1Qcy-2019, and IEEE Std 802.1Qcx-2020)*, pp. 1–151. DOI: 10.1109/IEEESTD.2020.9253013.

[IEEE 802.1Qci] "IEEE Standard for Local and Metropolitan Area Networks–Bridges and Bridged Networks–Amendment 28" (Sept. 2017). "IEEE Standard for Local and Metropolitan Area Networks–Bridges and Bridged Networks–Amendment 28: Per-Stream Filtering and Policing." In: *IEEE Std 802.1Qci-2017 (Amendment to IEEE Std 802.1Q-2014 as amended by IEEE Std 802.1Qca-2015, IEEE Std 802.1Qcd-2015, IEEE Std 802.1Q-2014/Cor 1-2015, IEEE Std 802.1Qbv-2015, IEEE Std 802.1Qbu-2016, and IEEE Std 802.1Qbz-2016)*, pp. 1–65. DOI: 10.1109/IEEESTD.2017.8064221.

[IEEE 802.1Qch] "IEEE Standard for Local and Metropolitan Area Networks–Bridges and Bridged Networks–Amendment 29" (June 2017). "IEEE Standard for Local and Metropolitan Area Networks–Bridges and Bridged Networks–Amendment 29: Cyclic Queuing and Forwarding." In: *IEEE 802.1Qch-2017 (Amendment to IEEE Std 802.1Q-2014 as amended by IEEE Std 802.1Qca-2015, IEEE Std 802.1Qcd(TM)-2015, IEEE Std 802.1Q-2014/Cor 1-2015, IEEE Std 802.1Qbv-2015, IEEE Std 802.1Qbu-2016, IEEE Std 802.1Qbz-2016, and IEEE Std 802.1Qci-2017)*, pp. 1–30. DOI: 10.1109/IEEESTD.2017.7961303.

[IEEE 802.1CB] "IEEE Standard for Local and Metropolitan Area Networks–Frame Replication and Elimination for Reliability" (Oct. 2017). In: *IEEE Std 802.1CB-2017*, pp. 1–102. DOI: 10.1109/IEEESTD.2017.8091139.

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

[8] *INET Framework - INET Framework* (2022). URL: https://inet.omnetpp.org/ (visited on 04/18/2022).

[9] *INET Framework for OMNEST/OMNeT++* (Apr. 14, 2022). INET Framework for OMNeT++. URL: https://github.com/inet-framework/inet/blob/95c53903478f96bbbded2079bb062c58 src/inet/clock/base/ClockBase.cc (visited on 04/18/2022).

[10] *INET Framework for OMNEST/OMNeT++* (Apr. 14, 2022). INET Framework for OMNeT++. URL: https://github.com/inet-framework/inet/blob/95c53903478f96bbbded2079bb062c58 src/inet/clock/model/OscillatorBasedClock.cc (visited on 04/18/2022).

[11] *Ipvs / Nesting* (2022). GitLab. URL: https://gitlab.com/ipvs/nesting (visited on 04/16/2022).

[ISO/IEC/IEEE 15288] "ISO/IEC/IEEE International Standard - Systems and Software Engineering – System Life Cycle Processes" (May 2015). In: *ISO/IEC/IEEE 15288 First edition 2015-05-15*, pp. 1–118. DOI: 10.1109/IEEESTD.2015.7106435.

[ITU G.810] ITU (1996). "Definitions and Terminology for Synchronization Networks." In: *ITU G.810*. URL: https://www.itu.int/rec/T-REC-G.810-199608-I/en (visited on 10/14/2019).

[ITU G.812] – (2004). "Timing Requirements of Slave Clocks Suitable for Use as Node Clocks in Synchronization Networks." In: *ITU G.812*. URL: https://www.itu.int/rec/T-REC-G.812-200406-I/en (visited on 10/23/2019).

[12] Le Boudec, Jean-Yves, director (Apr. 5, 2019). *An Introduction to Network Calculus*. URL: https://www.youtube.com/watch?v=ABQ327BTc_o (visited on 04/21/2022).

[13] Maile, Lisa, director (July 27, 2020). *Network Calculus for Time-Sensitive Networking: An Introduction*. URL: https://www.youtube.com/watch?v=5Y_wYyAWjrE (visited on 04/21/2022).

[RFC 5905] Martin, Jim, Jack Burbank, William Kasch, and Professor David L. Mills (June 2010). *Network Time Protocol Version 4: Protocol and Algorithms Specification*. Request for Comments RFC 5905. Internet Engineering Task Force. 110 pp. DOI: 10.17487/RFC5905. URL: https://datatracker.ietf.org/doc/rfc5905 (visited on 05/03/2022).

[14] Matthieu, Coudron (Tue Aug 25 07:05:35 PDT 2015). *[Ns-developers] [Gsoc 2015] Final Review - Clock Support & Mptcp implementation*. E-mail. URL: https://mailman.isi.edu/pipermail/ns-developers/2015-August/012898.html (visited on 04/12/2022).

[RFC 4737] Morton, Al, Gomathi Ramachandran, Stanislav Shalunov, Len Ciavattone, and Jerry Perser (Nov. 2006). *Packet Reordering Metrics*. Request for Comments RFC 4737. Internet Engineering Task Force. 45 pp. DOI: 10.17487/RFC4737. URL: https://datatracker.ietf.org/doc/rfc4737 (visited on 01/27/2022).

[15] *Multiple Inheritance* (Mar. 10, 2022). In: *Wikipedia*. URL: https://en.wikipedia.org/w/index.php?title=Multiple_inheritance&oldid=1076274054 (visited on 04/08/2022).

[16] *Ns-3 Network Simulator. Project Homepage* (2011–2020). ns-3. URL: https://www.nsnam.org/ (visited on 01/21/2020).

[17] *OMNeT++ Discrete Event Simulator* (2022). URL: https://omnetpp.org/ (visited on 04/10/2022).

[18] *OPNET Network Simulator* (2022). Opnet Projects. URL: https://opnetprojects.com/opnet-network-simulator/ (visited on 04/10/2022).

[IEEE P802.1DG] *P802.1DG – TSN Profile for Automotive In-Vehicle Ethernet Communications* (2022). URL: https://1.ieee802.org/tsn/802-1dg/ (visited on 01/01/2022).

[IEEE P802.1DP] *P802.1DP – TSN for Aerospace Onboard Ethernet Communications* — (2022). URL: https://1.ieee802.org/tsn/802-1dp/ (visited on 01/01/2022).

[ITU G.8261] Recommendation, ITUTG (2006). "8261/Y. 1361 Timing and Synchronization Aspects in Packet Networks." In: *International Telecommun. Union*. URL: https://www.itu.int/rec/T-REC-G.8261.

[19] SIGCOMM, ACM (2022). *SIGCOMM Networking Systems Award — Acm Sigcomm*. Cité dans chap ns-3. URL: https://www.sigcomm.org/content/sigcomm-networking-systems-award (visited on 04/07/2022).

[20] Thomas, Ludovic (2022). *THOMAS Ludovic / Xtfa*. ISAE-SUPAERO. URL: `https://gitlab.isae-supaero.fr/l.thomas/xtfa` (visited on 06/22/2022).

[21] *Time-Sensitive Networking (TSN) Task Group — Task Group Website* (2021). URL: `https://1.ieee802.org/tsn/` (visited on 12/31/2021).

[INCOSE SE Handbook] Walden, David D., Garry J. Roedler, Kevin Forsberg, R. Douglas Hamelin, Thomas M. Shortell, and International Council on Systems Engineering, eds. (2015). *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*. Fourth edition. Hoboken, New Jersey: Wiley. ISBN: 978-1-118-99940-0.