Scalable Trajectory Approach for ensuring deterministic guarantees in large networks

Sara Medlej


UNIVERSITE PARIS-SUD

ECOLE DOCTORALE D'INFORMATIQUE DE PARIS-SUD

Laboratoire de Recherche en Informatique

DISCIPLINE : COMPUTER SCIENCE

DOCTORAL THESIS

defended on September, 26^{th} 2013

by

Sara MEDLEJ

Scalable Trajectory Approach to ensure deterministic guarantees in large networks

Jury members:

Reviewers:	Guy Pujolle, Université Pierre et Marie Curie	Professor
	Houda Labiod, Telecom ParisTech	Associate Professor
Examiners:	Hakima Chaouchi, Telecom SudParis	Professor
	Véronique Véque, Supelec	Professor
Supervisor	Khaldoun Al Agha, Université Paris-Sud	Professor
Supervisor	Steven Martin, Université Paris-Sud	Associate Professor

Acknowledgment

I owe my gratitude to all the people who have made this thesis possible with their help, support and contributions.

It is my pleasure to thank my supervisor, Dr. Steven Martin, for offering me this opportunity and helping me in every problem I encountered in this period. His good advice, support and knowledge have been invaluable on both academic and personal level for which I am extremely grateful.

I would like also to thank Prof. Khaldoun Al AGHA, for the insightful discussions and support in all stages of this thesis.

I would like to express my gratitude to the whole team P1A in EDF for their sympathy and helpful collaboration. I appreciate the valuable comments provided by Mr. Jean-Marie COTTIN, Mr. Denis TROGNON and Dr. Gaelle MARSAL.

I would like to express my deepest gratitude to the reviewers Dr. Houda LABIOD and Prof. Guy PUJOLLE for having the patience to read the dissertation and provide me with constructive comments. Many thanks go to all members of the jury, Prof. Hakima CHAOUCHI, Prof. Veronique VEQUE for coming to serve on my thesis committee, and for sparing their invaluable time reviewing the manuscript.

I would like to express my appreciations to all my colleagues in the LRI laboratory, Asma, Valeria, Roccio, Joseph, Soran, Reben, Hassan, Guangy, Kehao, Simon and

Thomas, who have enriched my graduate life in many ways. I have also appreciated the good time I have spent with my friends from the algorithmic and Database team. The last year has been less stressing thanks to Dr. Reza and the funny moments that we have shared. Finally, special thanks to my dearest friend Youghourta for all the fruitful discussions we had, and of course for allowing me to keep Uba for an additional year.

My stay in France wouldnt be so pleasant without the presence of all my Lebanese friends in Rennes, Troyes and Paris: Hadi, Youssef, Ali, Rima, Sara J., Abbas, Rihame, Hussein, Moussa, Sara K., Sarab, Kassem, Rola and Mazen, each one of you has a special place in my heart. And of course, I cant forget Zeinab, Imane and Lana who were there every time I needed them; thank you girls for being who you are.

Lastly, I would like to thank my Sweetheart Hussein for his unconditional love, support, encouragement and unlimited kindness through all these years. I express my regards to my fiancé family for their kindness. My life would be really boring without the presence of my brother Hussein and his wife Hanan, my sister Rita and her husband Nizar, my nephews Ali, Hussein and Karim who took as a mission cheering me up. Finally, words cannot express my gratitude and love for my mom and dad: the best thing that happened to me is being your daughter. I am blessed for having you in my life.

Abstract

In critical real-time systems, any faulty behavior may endanger lives. Hence, system verification and validation is essential before their deployment. In fact, safety authorities ask to ensure deterministic guarantees. In this thesis, we are interested in offering temporal guarantees; in particular we need to prove that the end-to-end response time of every flow present in the network is bounded. This subject has been addressed for many years and several approaches have been developed. After a brief comparison between the existing approaches, the Trajectory Approach makes a good candidate due to the tightness of its offered bound. This method uses results established by the scheduling theory to derive an upper bound. The reasons leading to a pessimistic upper bound are investigated. Moreover, since the method must be applied on large networks, it is important to be able to give results in an acceptable time frame. Hence, a study of the method's scalability was carried out. Analysis shows that the complexity of the computation is due to both recursive and iterative processes. As the number of flows and switches increases, the total runtime required to compute the upper bound of every flow present in the network understudy grows rapidly. While based on the concept of the Trajectory Approach, we propose to compute an upper bound in a reduced time frame and without significant loss of precision. It is called the Scalable Trajectory Approach. After applying it to a network, the simulation results show that the total runtime was reduced from several days to a dozen of seconds.

Contents

List of Figures

List of Tables

Acronyms and notations

List of acronyms

AFDX	Avionic Full DupleX
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
DP	Dynamic Priority
EDF	Earliest Deadline First
ETA	Enhanced Trajectory Approach
EWCRT	Exact Worst-Case Response Time
FIFO	First In First Out
FP	Fixed Priority
MAC	Medium Access Control
MC	Model Checking
NC	Network Calculus
RM	Rate Monotonic
STA	Scalable Trajectory Approach
TA	Trajectory Approach

List of notations

τ_i	sporadic flow
C_i^h	processing time of a packet belonging to flow τ_i on node h
T_i	minimum arrival time of two successive packets of flow τ_i
J_i	activation jitter of flow τ_i
D_i	deadline of flow τ_i
L	Switching fabric delay including propagation delay, consulting the MAC address table and copying the frame into the output buffer
$W_{i,t}^q$	latest execution time of flow τ_i on node q
R_i	worst case response time (WCRT) of flow τ_i
$\mathcal{P}_i$	path of flow τ_i
$\|P_i\|$	length of the path of τ_i
$first_i$	first node in the path of flow τ_i
$last_i$	last node in the path of flow τ_i
$first_{i,j}$	first node in common between $\mathcal{P}_i$ and $\mathcal{P}_j$
$last_{i,j}$	last node in common between $\mathcal{P}_i$ and $\mathcal{P}_j$
$S_{max_j}^h$	maximum time put by a packet of flow τ_j to travel from its source to node h
$S_{min_j}^h$	minimum time put by a packet of flow τ_j to travel from its source to node h
M_i^h	minimum time put by a packet to travel from the source of flow τ_i to node h
$A_{i,j}$	Sum of the arrival jitters of flows τ_i and τ_j

Chapter 1

Introduction

Contents

First, in section 1.1, the context and the problematic are presented. Secondly, section 1.2 details the characteristics of the studied network. Thirdly, the property under study is invoked in section 1.3. Then, our contributions are listed in section 1.4. Finally, the outline dissertation is presented in section 1.5.

1.1 Context and Problematic

Engineering is human endeavor and thus it is subject to errors. Some engineering errors are neglectable, as when a new concrete building develops cracks. While, some errors seem humanly unforgettable, like when a bridge collapses resulting in the death of those who had taken its soundness for granted [1].

Ensuring that a system is safe is not a newly addressed subject. Almost four thousands years ago, a number of Babylonian law codes were collected in what has known to be the *Hammurabi Code*. Among nearly three hundred cuneiform inscriptions governing matters related to the status of women, drinking-houses, etc, there are those related to the safety of constructions.

In [2], authors illustrate mistakes in requirements and/or system verification and validation that has lead to famous failures. The Tacoma Narrows bridge, for example, was a scale up of an old design. In the design of the first one, the effect of the winds was not considered. Eventually, the bridge became unstable in the crosswinds and it collapsed four months after its opening. We can also list the Space shuttle challenger, the chernobyl nuclear power plant, Arian 5 missile. For other famous failures see [1], [3], [4], and [5].

Nowadays, the Ethernet has also been used in hard real-time systems such as in nuclear power plants, on board airplanes and so on. Accordingly, before deploying any system, it is necessary to ensure that a set of logic and temporal properties are

respected. This thesis focuses only on temporal property, particularly the end-to-end response time (or delay) of a packet across the network.

1.2 Characteristics of the studied network

The studied network is a full-duplex switched-Ethernet. It is composed of one hundred identical switches connected in a line topology (see Fig. 1.1). Each switch is connected to a single node which can be either a sensor node or an actuator.

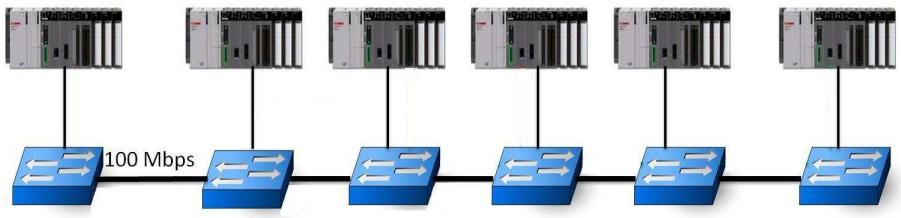


Figure 1.1: The network under study.

The used switches are 8 ports Ethernet switches implementing store and forward strategy. As illustrated in Fig. 1.2, for each output port a 1MByte buffer is dedicated. The serving policy used in this kind of switches is the First In First Out (FIFO) policy which means that packets are scheduled according to their arrival time. Each switch keeps updated a so called bridge forwarding table. It associates to each MAC address the port on which the packet must be sent to reach its destination. Moreover, the switching delay - including consulting the MAC table, copying a packet into the corresponding output buffer, etc.- is considered constant and is equal to $3\mu s$.

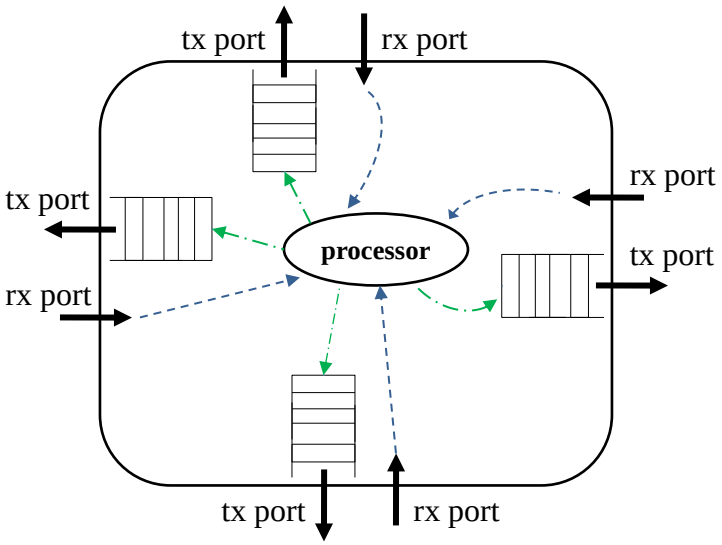


Figure 1.2: Buffering inside a switch

1.3 Safety property

In order to be certified as secure, the deployed network has to verify several properties listed in [6]. In our work, we concentrate on the first property: *In normal functioning, every transmitted message reaches the right destination in a bounded time*. We assume that each message reaches the correct destination. Hence our work consists of proving that the message end-to-end response time (or delay) is bounded. It is important to show that this bound does not exceed a specific deadline. Therefore, to avoid overdimensioning the network, the upper bound should be as close as possible to the exact worst-case response time.

1.4 Contributions

Our work consists of focusing on a method called the Trajectory Approach that is recently used for upper bounding the end-to-end response time and that showed its efficiency when compared to the Network Calculus - a method traditionally used for the same purpose [7].
In the following, we list our main contributions:

1. **Identifying the main sources of pessimism for FIFO policy**
   The upper bound obtained using this method is sometimes not precise. We show on small configurations how other flows affect the tightness of this bound. We have noticed that when flows leave the path of the studied one or the path of a flow interacting directly or indirectly with the studied one, the computed bound may be pessimistic. Results were published in [8].

2. **Proposing a scalable Trajectory Approach**
   The second parameter studied when applying the adopted method is its scalability. Indeed, it must be applied on large industrial networks composed of hundreds of switches and where a large amount of messages are exchanged. As the original version of the method fails to offer results in these networks, we propose to compute an upper bound in a reduced time frame and without significant loss in the precision of the upper bound. The obtained results were submitted to RTSS'13 (Real Time System Symposium 2013) and RTNS'13 (Real-Time Systems and Software 2013)

1.5 Dissertation outline

This thesis is composed of five chapters in which the problematic is first exposed, then our contributions are presented and finally the proposed approach is evaluated on large network.

In this chapter, we presented the studied problematic: worst-case timing analysis. We are interested in estimating the worst-case upper bound of the end-to-end response time (or delay) of a flow.

Chapter 2 presents some methods existing in the state of the art useful when tackling the problem of ensuring the determinism of a network. After explaining their principles, the advantages and drawbacks are listed. As a result, we conclude that the Trajectory Approach makes a good candidate.

In chapter 3, we discuss the limitations of applying the adopted approach on a large switched-Ethernet network. The flows in this case are scheduled using FIFO serving policy. The investigated properties are the precision and the scalability of the approach.

Chapter 4 is dedicated for ensuring the scalability of the approach. We propose a computation based on the Trajectory Approach allowing to determine an upper bound of the end-to-end response time but in a smaller time frame.

Finally, chapter 5 holds the conclusion and perspectives.

Chapter 2

Temporal guarantees

Contents

2.1 Introduction

As described before, we need to prove that every flow in the studied network respects its deadline which means that the end-to-end response time does not exceed the corresponding deadline. In section 2.2, we briefly explore existing methods that are used to study the temporal behavior of a real-time system. Then, the advantages and drawbacks of these methods are listed in 2.3. Finally, the conclusion is presented in section 2.4.

2.2 Timing analysis

Ethernet is widely deployed in local area networking solutions at home and offices. This huge spread of the Ethernet is due to its fast and easy way of installation. In addition, this technology has lately seen an increase in the speed (1Gbps and up to 100Gbps). Moreover, most computerized devices come with an Ethernet interface. For all these reasons, the Ethernet technology has found its way to the industrial sector. It is now being used in even most critical domains such as on board airplanes, in nuclear plant, etc.

However, the major issue confronted when using Ethernet in distributed real-time applications is its stochastic property that is generated by the collision recovery mechanism. To get rid of this problem, hub-based infrastructure was simply replaced by Ethernet switches in FDX (Full Duplex). Switched Ethernet creates point-to-point connections between communicating entities, therefore eliminating any kind of collisions [9]. By using the switched Ethernet, the problem resulting form sharing the medium was overcame. Nevertheless, a new problem arises: since packets are now competing over the switch's resources (i.e. output buffers), the switches must

be well dimensionned to avoid any possible overflow.

In critical real-time systems such as those used in the avionic, the nuclear or the chemical domains, any faulty behavior may endanger lives. Hence system verification and validation is essential before its deployment. In fact, safety authorities ask, before delivering their certificate, to ensure deterministic guarantees. Those guarantees usually mean that the underlying network must ensure bounded end-to-end response time, bounded (or null) jitter, reliability (with respect to corruption, loss due to overflow or duplication of messages). In our study, we are only interested in offering temporal guarantees; in particular we need to prove that the end-to-end response time of every flow, present in the network, is bounded. This subject has been addressed for many years and several approaches have been developed tackling it. In the literature, simulation and/or testing, formal verification methods especially Model Checking and Network Calculus were used to estimate end-to-end response time in complex systems. In addition, another method called the Trajectory Approach addresses the same problem.

In the following, we describe the principle on which these methods are based and list their advantages as well as their drawbacks.

2.2.1 Simulation-based approach

Simulation and *testing* are both conducted before deploying the system in the field. The difference between the two of them is that when simulating we are interacting with software that imitates the behavior of the system under study. Where, testing deals with the product itself meaning that we are interacting with a hardware. For instance, to check the performance of an electronic design, we can use Pspice [10] to emulate the behavior of the system; in this case we are performing a simulation.

While if we are dealing with the electrical board itself, we are conducting tests. The common point in both cases is that an input is inserted into the system and the system behavior, represented by the output, is observed. WebNSM Simulation Tool Kit [11] is an example of network simulator used for several purposes such as automated network simulation, configuration of device values and simulation types for performance testing. Another tool is the famous Matlab/simulink software [12] that allows us to evaluate the temporal behavior of systems.

Being a costless method, simulation and testing have been used for a long time. In contrast, the main disadvantage of these methods is the fact that it is impossible to check all possible events of the system and that sometimes rare event can be missed when performing the testing and/or the simulation. Consequently, the determined bound offered by these methods is not very accurate and is sometimes below the real value [13]. However, when asked to offer deterministic guarantees, underestimating the bound is not tolerated. In addition, the problem is characterized as NP-hard which means that to test all possible events, a very long period of time is required in order to obtain results and in most scenarios, this period may reach up to several years. For all these reasons, the simulation-based approaches fail to fulfill our requirements.

2.2.2 Analytical approach

Performing appropriate mathematical analysis contributes to the reliability and robustness of the design. Several mathematical approaches were developed allowing the evaluation of the network performance. Our interest is not to explore all of them, but to introduce the most used ones and to choose the best candidate. In the following sections, we briefly introduce the *Model Checking*, a formal method, then we

talk about the *Network Calculus* and finally the *Trajectory Approach* is presented.

2.2.2.1 Model Checking

Performance evaluation aims at predicting system behavior in terms of delay, throughput, etc. This kind of evaluation addresses quantitative questions. However, traditional formal verification answers qualitative questions. Formal verification addresses problems related to safety, and liveness (i.e. does a packet reach the correct destination?). If considering the elevator system for example, the safety property is satisfied if the elevator car stops whenever the door opens and the liveness property is satisfied if the elevators never get stuck between doors. On the other hand, Model Checking [14], for instance, allows obtaining quantitive results. Model Checking (MC) is based on timed automata and was presented by Alur and Dill in [15]. It offers the possibility of analyzing a system while taking into consideration the time. The worst-case scenario and the corresponding end-to-end response time (or delay) is obtained [16], [17]. MC has been used in many real applications including electrical circuits, communication protocols, digital controllers, etc.

Model checking consists of three tasks which are modeling, specification and finally verification. Fig. 2.1 shows the concept on which this method is based. The approach requires a model of the studied system and a desired property to be verified. After modeling the system and formalizing the property, the Model Checking tools exhaustively checks if the property is satisfied in each state of the system model. If an error occurs, the approach generates the exact scenario that have caused this error. Hence, providing an evidence that the system is faulty and need to be revised [18]. However, if the property is satisfied by the system, meaning no error was found, then the developer can refine his model and retest it. UPPAAL [19], BLAST

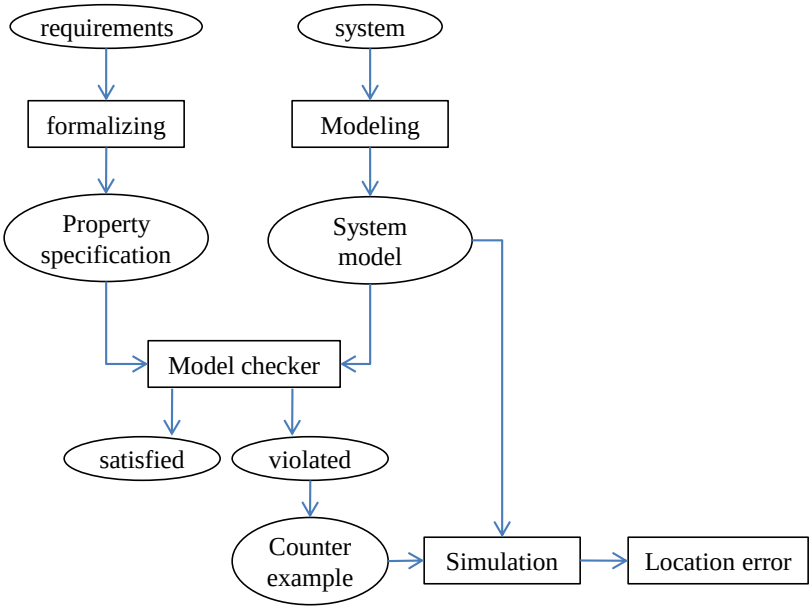


Figure 2.1: Model checking concept

model checker [20], SPIN model checker [21] are examples of model checking tools. It is necessary to be careful when formalizing the property desired to be satisfied by the system. As mentioned in [14], the Model Checking is able to identify if a system satisfies a given property but it is impossible to check whether the given specification covers all properties that the system must satisfy. Thus, the completeness of the specification has an important impact on the results.

Moreover, the size of the model affects the performance of the Model Checking which was shown in [14]. The authors derive three models for the same architecture and check a temporal logic for each model using Model Checking. The architecture under inspection is composed of 12 interacting components. In the first model, the behavior of all components is detailed. The second model is a simplified version of the first model where components that do not affect directly the temporal logic are eliminated and the behavior of the remaining components is also simplified. The

third model is obtained by applying the state space approximation on the second model. For the first model, the system explodes and no result was computed. When using the second model, 28 hours were necessary to compute the result while using the third model reduces the computing time to only one second. There is no doubt that the main challenge confronted when using Model Checking is the state space explosion problem. This problem arises when the global states are enormous which can happen in a system having a large number of components interacting. Such a problem makes the approach incapable of providing solution when dealing for example with large industrial networks similar to our case.

In the last years, the state space explosion problem has attracted many researchers attempting to overcome it and several solutions were proposed. According to [22], these solutions can be divided into three categories which we describe succinctly.
The first category is based on automata theory which consists of three steps. Step one consists of converting the modeled system into the buchi automaton A which is an extension of a finite state automaton to infinite inputs. The second step consists of translating the negation of the specification into an automaton S. In the last step the emptiness of the intersection between the Kripke structure M and the automaton S is checked. If the intersection between M and S is not empty, a counter example is reported. As defined in [23], a kripke structure is a directed graph whose vertices are labeled by a set of atomic propositions. Vertices represent states of the system and edges represent transitions between states. The kripke structure M over a set of atomic propositions A is a tuple $M = (S, R, L, I)$ such that S is the set of states, $R \subseteq S^2$ is the set of transitions, $I \subseteq S$ is the initial states which should not be empty and $L : S \to 2^A$ labels each state by a set of atomic propositions. The kripke structure can be also viewed as an automaton.

The second category is based on symbolic verification in which rather than explicitly constructing the kripke structure, boolean functions representing respectively transition relations and the set of states are computed. Then model checking algorithm is applied on these functions. Since boolean functions are exponentially smaller than the explicit representation, symbolic verification theoretically alleviates the state space explosion problem [23], [22]. The main ingredient in symbolic verification is the Binary Decision Diagram BDD. Given a boolean function, binary decision tree is constructed on condition that along a path from root to leaf, variables are listed in the same order and each variable appears just once. Reducing the size of a BDD can be achieved by applying two rules:

- merging all duplicate nodes
- removing nodes if their branches end up on the same child node.

BDD in its current form is called a reduced and ordered binary decision diagram (ROBDD). It is clear that the complexity of the symbolic model depends on the size of the BDD. However, classic information theory argument shows that only a small fraction of all finite kripke structure can be exponentially compressed [24]. Similarly, the same limitation applies as well on the BDD. Moreover, practical experiments show that the performance of symbolic verification methods is highly unpredictable. This phenomena can be partially explained by complexity theory which states that BDD representation does not improve worst-case complexity.
The third category, called alternative methods, includes methods such as abstraction and symmetry. The abstraction concept consists of partitioning the states of Kripke structure into clusters and treating clusters as new abstract states. Applying this method reduces the model which makes it fit into memory. If the abstract model

satisfies the property, we deduce that the original Kripke structure also satisfies the property specification.

2.2.2.2 Network Calculus

Network Calculus is based on the min-plus algebra and is used to evaluate deterministic bounds for queuing systems encountered in communication systems; allowing thus to offer deterministic guarantees for a flow or a group of flows. It was first presented by Cruz in [25], [26]. The concept of the service curve has been formalized by Cruz, Sariowan, Cruz and Polyzos [27],[28], Argual and Rajan [29], Chang [30], [31], Le Boudec [32], and Agrawal, Cruz, Okino and Rajan [29] toward the general framework known as network calculus today.

Based on node's arrival and service curves, Network Calculus derives bounds on the end-to-end delay and backlog. In the following, we explain the notion of arrival and service curves.

Data flow can be described as a wide-sense increasing function $R(t)$ which represents the number of bits seen in an interval of time $[0, t]$. Let us consider a black box S which can be a simple buffer, a complex node or even an entire network. S receives a data flow represented by the function $R(t)$ at its input. S delivers the data on its output after a variable delay. The output data is characterized by a cumulative function $R^*(t)$. Several policies can be implemented by S to process incoming data. For example a packet is processed as soon as its first bit is received by the system or until the entire packet is received, etc. Each flow R is constrained by an arrival curve $\alpha(t)$ if and only if $R(t) - R(s) \leq \alpha(t - s)$ for all $s \leq t$. It means that the number of bits arriving between time s and t is at most $\alpha(t - s)$. The arrival curve $\alpha(t)$ is a wide sense increasing function. The most famous arrival curves are the leaky bucket

and the generic cell algorithm; the latter corresponds to a stair function. The leaky bucket curve, depicted in Fig. 2.2, is represented by the formula $\alpha(t) = b + rt$ where r is the rate and b is the burst size.

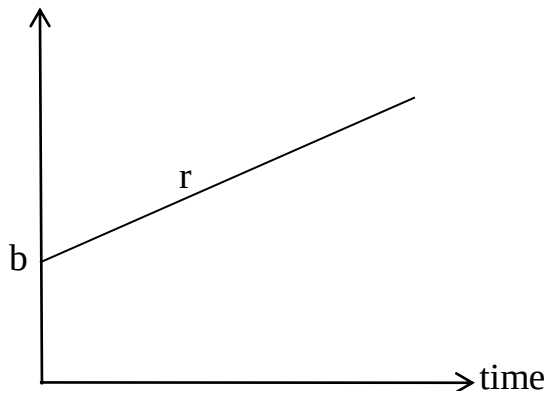


Figure 2.2: leaky bucket arrival curve

To offer guarantees to a data flow, the data must be declared conformant in other words it must not lead to a buffer overflow. We are considering for instance a flow constrained by a leaky bucket arrival curve. The flow R is said to be conformant if the amount of poured data into the bucket is equal to the amount of data exiting in the bucket. We say that the system S offers a service curve $\beta(t)$ if and only if $R^* \geq (R \otimes \beta)(t)$ where $(R \otimes \beta)(t) := \inf_{0 \leq \tau \leq t}\{R(\tau) + \beta(t - \tau)\}$ and $\beta(t)$ is a wide sense increasing function. Based on arrival and service curves, Network Calculus can derive three parameters which are the delay, the backlog and the output flow.

1. The backlog is constrained by vertical deviation between arrival curve and the service curve.

2. The delay is constrained by horizontal delay between arrival and service curves

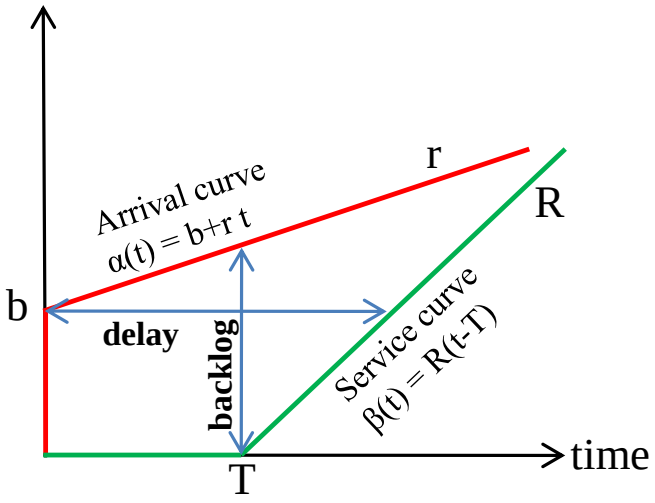


Figure 2.3: The backlog and the delay derived using Network Calculus

3. Output flow is constrained by the arrival curve $\alpha^*(t) = \alpha(t) \otimes \beta(t)$

An example on how to derive the delay and the backlog is depicted in Fig. 2.3. If $r \leq R$, then the delay bound is $d = T + b/R$, otherwise the delay is infinite.

It is obvious that the precision of the end-to-end delay bound depends on the accuracy of the arrival and the service curves. The authors in [33] consider a system composed of two nodes $N1$ and $N2$ and then compute the delay bound. Each of the nodes has, respectively, $\alpha_1(t)$ and $\alpha_2(t)$ as arrival curves and $\beta_1(t)$ and $\beta_2(t)$ as service curves. Arrival curves are constrained by leaky buckets. The delay D_1 introduced on the first node N_1 is equal to $\frac{b}{R_1} + T_1$ and the output flow on N_1 is constrained by $\alpha_1^*(t) = r(t + T_1) + b$. The output flow of node N_1 is the input flow of the node N_2, thus the delay experienced on the node N_2 is $D_2 = \frac{b+rT_1}{R_2} + T_2$. The delay bound of the entire network is equal to the sum of delays on each node $D_1 + D_2 = \frac{b}{R_1} + T1 + \frac{b+r \cdot T_1}{R_2} + T_2$. Hence, applying Network Calculus sequentially on

a chain of nodes yields in a high delay bound and the obtained bound is pessimistic. To improve the delay bound obtained using Network Calculus, the authors in [33] propose to consider a service curve of the entire system instead of applying Network Calculus on each node. The global service curve is the result of the convolution of service curves offered by each node of the system. Now going back to the previous example, computing delay bound gives a result equal to $\frac{b}{R_0} + T_0$ where $R_0 = \min\{R\}$ and $T_0 = \sum T_i$. Although using a global service curve, representing the service offered by the entire network, improves the computed end-to-end delay bound, it is not always easy to derive the global service curve. A survey on the service curves was conducted in [34].

A set of software implementing this method are available such as the CyNC toolbox [35], the RTC toolbox [36] [37], the DISCO network calculator [38], the COINC toolbox [39], Deborah [40] and finally the commercial SymTA/S toolbox [41].

2.2.2.3 Trajectory Approach (TA)

The approach has been developed in [42] for non-preemptive scheduling. It allows computing the upper bound of any flow scheduled using either Fixed Priority algorithms (FP), or Dynamic Priority algorithms (DP), or a combination of both. For FP algorithms, the packets belonging to the same flow have all the same priority level; while for DP algorithms the packet priority depends of its activation time. For instance, *Rate Monotonic* RM and *deadline monotonic* DM are fixed priority algorithms. On one hand, the priority assigned to a flow by RM is inversely proportional to its period. Thus, the flow with the shortest period has the highest priority. On the other hand, the priority assigned to a flow by DM in inversely proportional to its

relative deadline. Moreover, two of the most common used dynamic priority scheduling algorithms are the FIFO (First In First Out) and the EDF (Earliest Deadline First). At any time, EDF executes among the activated packets, those whose absolute deadline is earliest (note that the absolute deadline is equal to the activation time plus the relative deadline).

In [43], authors formalized the approach for FIFO scheduling policy and in [44] it was formalized for FP/FIFO. In this latter case, packets are first served according to their fixed priority, then those having the same fixed priority are scheduled according to their arrival time.

In the following, we present the traffic model used by the approach as well as the notations, then TA's concept is described.

Traffic model

A set of sporadic flows are exchanged across the studied network. A sporadic flow denoted τ_i is defined by the following tuple (C_i^h, T_i, D_i, J_i) such that:

- C_i^h is the processing time of any packet belonging to τ_i on node h,
- T_i is the minimum inter-arrival time between two successive packets of flow τ_i,
- D_i is the flow's deadline. The response time of a packet generated at time t should not exceed $t + D_i$.
- J_i is the release jitter of flow τ_i.

Moreover, each flow τ_i is processed on a set of nodes called path $\mathcal{P}_i$. For instance, the path of τ_1 in Fig. 2.4 is $\mathcal{P}_1 = \{N_1, SW_1, SW_2, SW_3, N_3\}$ and that of τ_2 is $\mathcal{P}_2 = \{N_2, SW_2, SW_3, SW_4, N_4\}$.

Moreover, we consider neither network failures nor packet losses. In addition, The

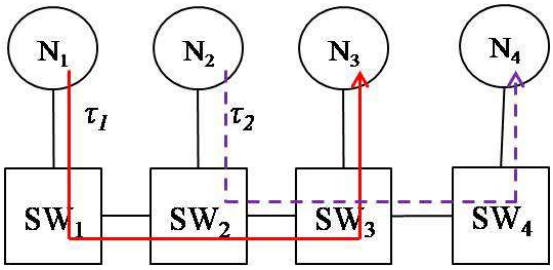


Figure 2.4: Example illustrating the path of a flow

switching delay (including consulting the MAC table, copying a packet into the corresponding output buffer, etc.) is respectively lower and upper bounded by L_{min} and L_{max}.

Mapping TA and traffic model

The mapping between the traffic model used by the Trajectory Approach and the studied network is illustrated in Fig. 2.5. The end-to-end response time of a flow τ_i is composed of the following:

- time spent by a packet denoted p in the buffer of the switch. It depends on the priority level of the packet and a_p^h its arrival time on node h. It is represented by the light gray period.
- packet processing time (or the transmission time over the link). $C_i^h = s/R$ where s is the packet size and R is the node data rate. If all nodes have the same data rate then the processing time of a packet is the same on all nodes ($C_i^h = C_i$). It is presented by the dark gray period.
- and finally, the switch introduces additional delay (or latency); it is called switching delay. In our case, it is equal to $L = 3\mu s$ and is represented by white

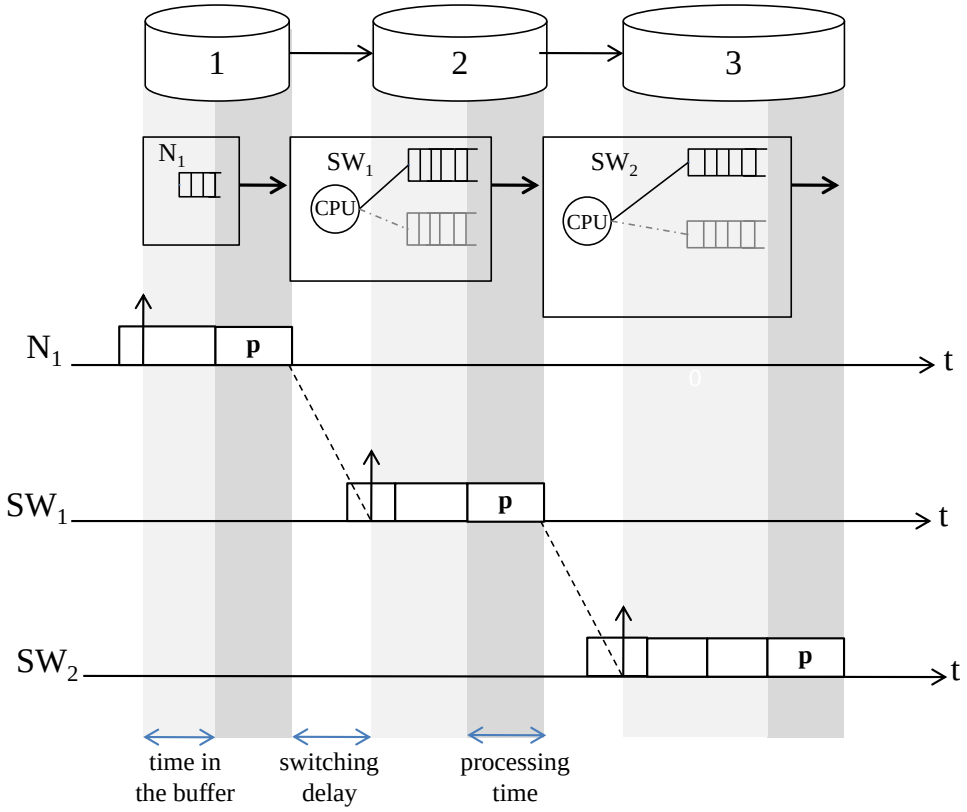


Figure 2.5: Illustration of the delays experienced by packet p

period.

Definitions and notation

The concept of the Trajectory Approach is based on the notion of the busy periods. Before explaining it, we illustrate first the notion of idle instants and busy periods. We also present some notations that are useful to understand the approach.

Definition 2.1. *An instant t_0 is said to be idle of level $\mathcal{L}$ if all packets activated*

before t_0 with a priority level higher or equal to $\mathcal{L}$ have been processed before t_0.

Definition 2.2. *A busy period of level $\mathcal{L}$ is an interval of time $[t_1, t_2[$ such that t_1 and t_2 are two idle instants of level $\mathcal{L}$ and there is no other idle instant in this interval.*

Fig. 2.6 illustrates the notions of idle instant and busy period. At time t_0, three packets p_1, p_2 and p_3 are activated, then at time t_1 another packet p_4 is activated. However, at this time (i.e. instant t_1) the execution of p_2, which was started before t_1, is not finished yet. This means that t_1 is not an idle instant. On the other hand, the execution of packets activated before t_2 have finished. We can deduce that t_2 is an idle instant. Hence, the interval $[t_0, t_2[$ forms a busy period.

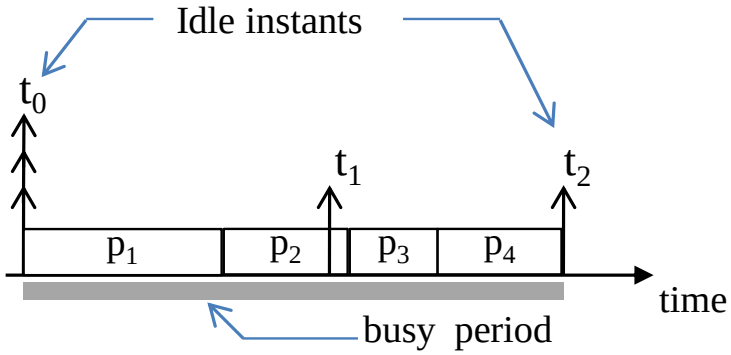


Figure 2.6: Example illustrating the notion of idle instants and busy period.

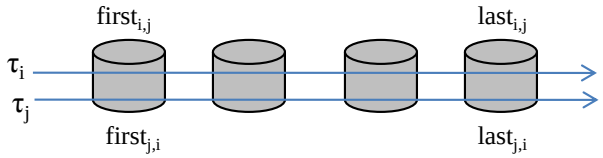
In addition, let τ_i, $i \in [1, n]$ be a sporadic flow following a path $\mathcal{P}_i$ and which is characterized by a priority P_i. Three sets are defined in [42]:

- $hp_i = \{j \in [1, n], P_j > P_i\}$ is the set of flows having a fixed priority strictly higher than that of flow τ_i;

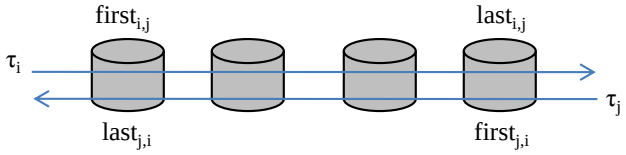
- $sp_i = \{j \in [1,n], j \neq i, P_j = P_i\}$ is the set of flows having a fixed priority equal to that of flow τ_i. Flow τ_i is not included in this set.

- $lp_i = \{j \in [1,n], P_j < P_i\}$ is the set of flows having a fixed priority strictly lower than that of flow τ_i.

Finally, let us consider two flows τ_i and τ_j that share a part of their paths. The notation $first_i$ (respectively $last_i$) denotes the source (respectively the destination) of flow τ_i. The first common node between the paths of τ_i and τ_j is denoted $first_{i,j}$ and the last common node is denoted $last_{i,j}$. The notion of $first_{i,j}$ and $last_{i,j}$ is illustrated in Fig. 2.7 when flows have a) the same direction b) reverse direction.



(a) same direction



(b) reverse direction

Figure 2.7: Illustration of $first_{i,j}$, $first_{j,i}$, $last_{i,j}$ and $last_{j,i}$ notations

TA concept

The Trajectory Approach is based on the analysis of the worst-case scenario experienced by a packet p of a sporadic flow τ_i along its path denoted $\mathcal{P}_i$. Traditional methods like the Holistic approach [45] considers the worst-case scenario on every

visited node, thus yielding pessimistic upper bounds [46]. Its objective is to compute an upper bound denoted R_i of the worst-case response time.

We consider that the path of the studied flow is numbered from 1 to q.

Let $r_{i,t}$ be the response time of packet p activated at time t on its source (i.e. node 1) and $W^q_{i,t}$ its latest execution time on the last visited node (i.e. node q).

As shown in Fig. 2.8, $r_{i,t}$ is equal to the latest execution time minus the packet activation time t plus the packet processing time C^q_i.

$$r_{i,t} = W^q_{i,t} - t + C^q_i \tag{2.1}$$

Let R_i be the upper bound of the worst-case response time of flow τ_i.

To determine the upper bound of the worst-case response time of a flow, it suffices to retain the maximum value of $r_{i,t}$ such that $t \geq -J_i$.

$$R_i = \max_{t \geq -J_i} \{r_{i,t}\} = \max_{t \geq -J_i} \left\{W^q_{i,t} - t + C^q_i\right\} \tag{2.2}$$

Hence, the main objective is to determine the expression of $W^q_{i,t}$.

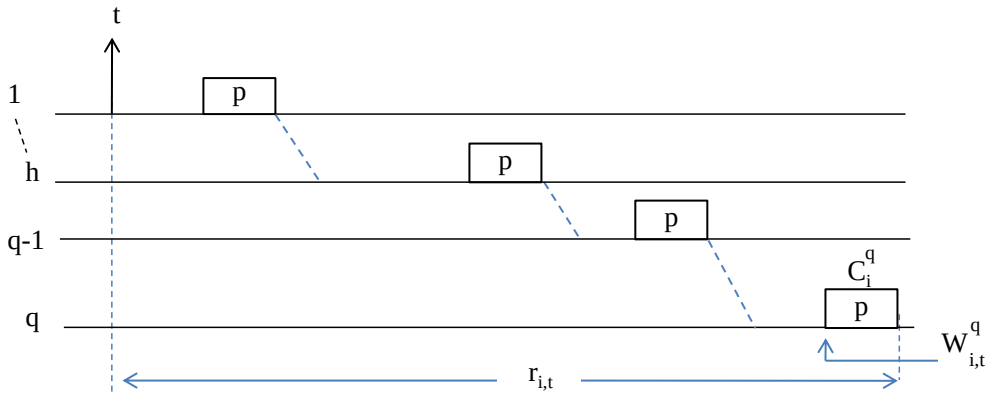


Figure 2.8: Response time of packet p activated at time t.

To achieve this goal, TA does not proceed like other traditional methods that start their analysis from the source and forward until reaching the destination. In fact, TA identifies the set of packets that postpone p and impact its response time on its visited nodes. To do so, it starts from the last node and goes backward until reaching the source. On the destination, the set of packets postponing the execution of the studied one are identified and form a busy period bp^q. In this busy period, two packets are defined: $f(q)$ and $p(q-1)$. $f(q)$ is the first packet of this period; it does not necessarily come from the previous node, i.e. node $q-1$. The packet $p(q-1)$ is the first packet of this busy period such that it was processed on node $q-1$.
Then on node $q-1$, packet $p(q-1)$ is located and the set of packets postponing its execution are identified. This packet $p(q-1)$ belongs to the busy periods of two successive nodes and is called *junction packet*. Next, packets $f(q-1)$ and $p(q-2)$ are located. Then, the same analysis occurs on the previous node (i.e node $q-2$) and so on, until determining the busy period on the source of the studied flow.
In brief, each busy period is composed of:

- packets having a priority higher than that of packet p,
- packets having the same priority level of packet p.

Finally, on each node h such that $h \in \mathcal{P}_i$, a single packet of a priority level lower than that of p can delay its execution. For this to be true, the execution of this packet must have been started before the arrival of packet $f(h)$ on the considered node.
Fig. 2.9 illustrates the concept of the Trajectory Approach.

As formalized in [47], the sum of these periods added to the total switching delay allows establishing the expression of the latest execution time of p. As said before, once computing $W_{i,t}^q$, obtaining the upper bound of the worst-case response time of a flow is quite simple. All we need to do is to recursively compute the response time

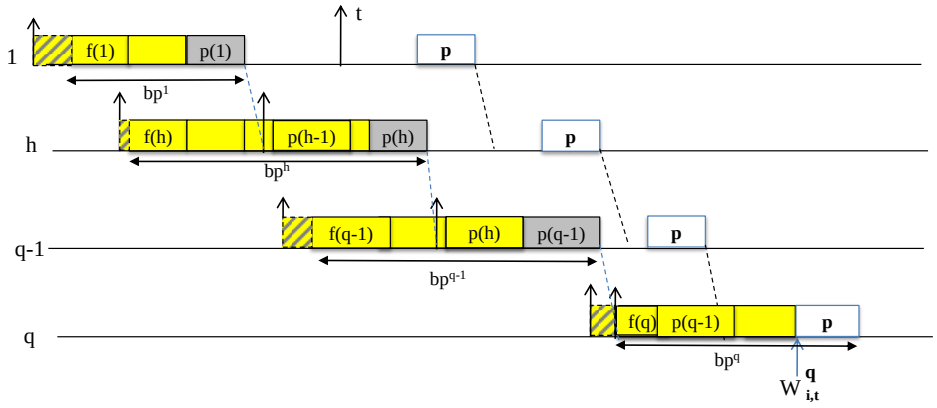


Figure 2.9: Trajectory Approach busy period decomposition

of packet p at each time t. Finally, the maximum value obtained is the upper bound of the studied flow.

Furthermore, it was proved in [42] that, except for the junction packet, a packet cannot belong to two different busy periods.

Given these facts, the expression of the latest execution time $W_{i,t}^{q}$ consists of three parts:

- the total switching delay which is equal to $(q-1)L_{max}$,
- the non-preemptive delay, denoted $\delta_i^{1,q}$, produced by the packets of lower priority level than p. On each node, only a single packet of this priority level can postpone the execution of the studied packet. The delay introduced by these packets is equal to $\sum_{h=1}^{q} \max\{0, \max_{j \in lp_i \bigcup spi(t)}\{C_j^h\} - 1\}$,
- the delay of packets having a priority level higher or equal to that of p. It can be expressed as follows: $X_{i,t} = \sum_{h=1}^{q} \sum_{g=f(h)}^{f(f+1)} C_{\tau(g)}^{h} - C_i^q$. The processing time of τ_i is subtracted since we are interested here in the expression of the latest execution time.

In summary, the expression of $W_{i,t}^{q}$ becomes:

$$W_{i,t}^{q} = (q-1)L_{max} + \delta_i^{1,q} + \sum_{h=1}^{q} \sum_{g=f(h)}^{f(h+1)} C_{\tau(g)}^{h} - C_i^{q} \tag{2.3}$$

We recall that junction packets are those processed at the end of the busy period on node h such that $h \neq q$ and at the beginning of the busy period on the next node $h+1$. Since we are interested in a worst-case analysis, then these packets should generate the largest delay on these nodes. Hence, the delay incurred is equal to $\sum_{h=1}^{q-1} \max_{j \in hp_i \bigcup sp_i} C_j^h$.

In order to assess the delay introduced by packets belonging to the busy periods, TA determines the largest interval of time in which if packets are activated on their sources, they will postpone the execution of the studied flow. This interval is called *generation interval*.

First, the delay incurred by packets of priority higher than that of packet p is evaluated. For these packets to delay the execution of the studied packet, they have to belong to one of the busy periods between the first common node $first_{i,j}$ and the last common one $last_{i,j}$. These packets postpone p if they arrive on its last common node (i.e. node $last_{i,j}$) at most before the beginning of its execution (i.e. $W_{i,t}^{last_{i,j}}$). Hence, they must be activated on their source $first_j$ at most before $W_{i,t}^{last_{i,j}} - S_{min_j}^{last_{i,j}}$ where $S_{min_j}^{h}$ is the minimum arrival time on node h. Moreover, packets of flow τ_j, $j \in hp_i$, belong to one of the busy periods if they arrive on the first common node $first_{i,j}$ at least after $a_{f(first_{i,j})}^{first_{i,j}}$ the arrival time of the first packet processed in the busy period. Thus, these packets must be generated on their sources at least after $a_{f(first_{i,j})}^{first_{i,j}} - S_{max_j}^{first_{i,j}} - J_j$. Similarly to $S_{min_j}^{h}$, $S_{max_j}^{h}$ is the maximum arrival time of a packet of τ_j on node h. Now, $a_{f(first_{i,j})}^{first_{i,j}}$ can be lower bounded by considering the minimum switching delay and counting the smallest packets processed on the nodes, starting from $first_i$ until reaching node $first_{i,j}$ (see Eq. (2.4)). Fig. 2.10 illustrates

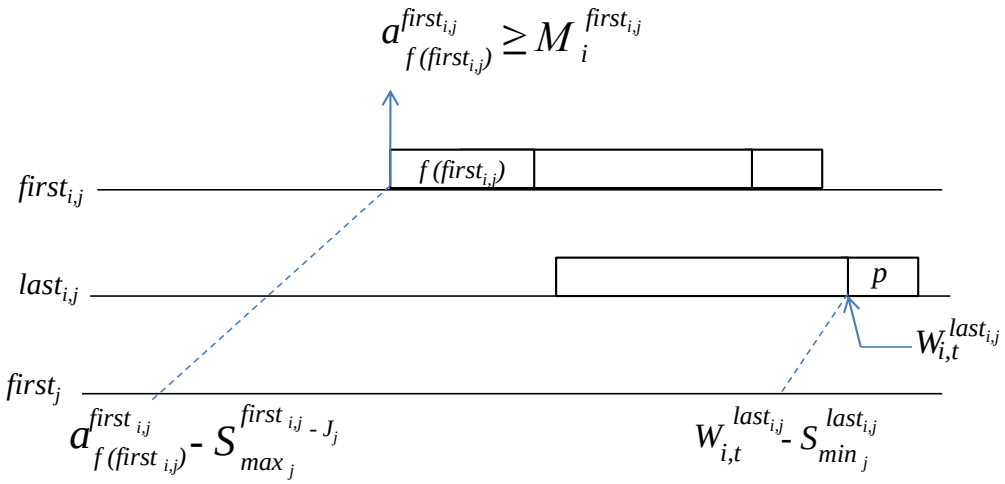


Figure 2.10: Generation interval of packets having a priority level higher than that of p

the length of the generation interval of higher priority packets.

$$a^{first_{i,j}}_{f(first_{i,j})} \geq M_i^{first_{i,j}} = \sum_{h'=first_i}^{first_{i,j}} \left(\min_{j \in hp_i \bigcup sp_i} \{C_j\} + L_{min} \right) \tag{2.4}$$

Secondly, flows having the same priority level as p postpone its execution, if on their first common node $first_{i,j}$ they arrive before $a_p^{first_{i,j}}$. This means that they should be generated on their source nodes at most before $a_p^{first_{i,j}} - S_{min_j}^{first_{i,j}}$. Besides, the arrival time of packet p on $first_{i,j}$ is smaller than the packet activation time t added to $S_{max_i}^{first_{i,j}}$ the maximum arrival time of flow τ_i on node $first_{i,j}$. Furthermore, these packets belongs to one of the busy periods if, on their first common node, they arrive at least after the arrival instant of $f(first_{i,j})$. Therefore, these packets must be generated on their sources at least after $a^{first_{i,j}}_{f(first_{i,j})} - S_{max_j}^{first_{i,j}} - J_j$. Fig. 2.11 illustrates the length of the generation interval of packets of the same priority level than p.

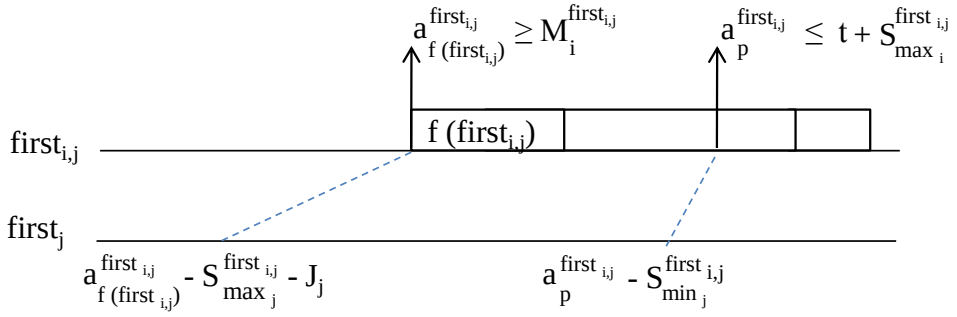


Figure 2.11: Generation interval of packets having the same priority level as packet p

$$a_p^{first_{i,j}} = t + S_{max_i}^{first_{i,j}} \tag{2.5}$$

Moreover, the delay incurred by packets belonging to flow τ_j activated in an interval $[a, b]$ is equal to $(1 + \left\lfloor \frac{b-a}{T_j} \right\rfloor) C_j$. Consequently, the delay incurred by packets of higher or similar priority can be deduced. To conclude, packets of τ_i are processed before packet p if they are activated before t. Hence, the delay incurred by these packets is equal to $\left(1 + \left\lfloor \frac{t}{T_i} \right\rfloor\right) C_i$.
Table 2.2 summarizes the interval of generation of packets postponing the studied flow so as their incurred delay.

flow	generation interval
τ_i	$[0, t]$
$j \in hp_i$	$\left[M_i^{first_{i,j}} - S_{max_j}^{first_{i,j}} - J_j, W_{i,t}^q - S_{min_j}^q\right]$
$j \in sp_i$	$\left[M_i^{first_{i,j}} - S_{max_j}^{first_{i,j}} - J_j, t + S_{max_i}^{first_{i,j}} - S_{min_j}^{first_{i,j}}\right]$

Table 2.1: generation interval of flows postponing p

flow	total delay
τ_i	$\left(1+\left\lfloor\frac{t}{T_i}\right\rfloor\right)C_i$
$j \in hp_i$	$\sum_j \left(1+\left\lfloor\frac{W^q_{i,t}-S^q_{min_j}+S^{first_{i,j}}_{max_j}-M^{first_{i,j}}_i+J_j}{T_j}\right\rfloor\right)C_j$
$j \in sp_i$	$\sum_j \left(1+\left\lfloor\frac{t+S^{first_{i,j}}_{max_j}-S^{first_{i,j}}_{min_j}+S^{first_{i,j}}_{max_i}-M^{first_{i,j}}_i+J_j}{T_j}\right\rfloor\right)C_j$
lp_i	$\delta_i^{1,q}=\sum_{h=1}^{q}\max\{0, max_{j\in lp_i \bigcup spi(t)}\{C_j\}-1\}$

Table 2.2: delay introduced by flows postponing p

Subsequently, the expression of the latest execution time becomes:

$$\begin{aligned} W^q_{i,t} &= (q-1)\cdot L_{max}+\delta_i^{1,q}+\sum_{h=1}^{q-1}\max_{j\in hp_i\bigcup sp_i}\{C_j\} \\ &+ \sum_{j\in hp_i}\left(1+\left\lfloor\frac{W^q_{i,t}-S^q_{min_j}+S^{first_{i,j}}_{max_j}-M^{first_{i,j}}_i+J_j}{T_j}\right\rfloor\right)C_j \\ &+ \sum_{j\in sp_i}\left(1+\left\lfloor\frac{t+S^{first_{i,j}}_{max_j}-S^{first_{i,j}}_{min_j}+S^{first_{i,j}}_{max_i}-M^{first_{i,j}}_i+J_j}{T_j}\right\rfloor\right)C_j \\ &+ \left(1+\left\lfloor\frac{t}{T_i}\right\rfloor\right)C_i-C_i \end{aligned}$$

Finally, the upper bound of flow τ_i can be expressed as follows.

$$\begin{aligned} R_i &= \max_{t\geq -J_i}\left\{(q-1)\cdot L_{max}+\delta_i^{1,q}+\sum_{h=1}^{q-1}\max_{j\in hp_i\bigcup sp_i}\{C_j\}\right. \\ &+ \sum_{j\in hp_i}\left(1+\left\lfloor\frac{W^q_{i,t}-S^q_{min_j}+S^{first_{i,j}}_{max_j}-M^{first_{i,j}}_i+J_j}{T_j}\right\rfloor\right)C_j \\ &+ \sum_{j\in sp_i}\left(1+\left\lfloor\frac{t+S^{first_{i,j}}_{max_j}-S^{first_{i,j}}_{min_j}+S^{first_{i,j}}_{max_i}-M^{first_{i,j}}_i+J_j}{T_j}\right\rfloor\right)C_j \\ &+ \left.\left(1+\left\lfloor\frac{t}{T_i}\right\rfloor\right)C_i-t\right\} \end{aligned}$$

The set of instants to test was reduced such that $-J_i \leq t \leq B_i^{slow}$, where:

$$B_i^{slow} = \sum_{j \in hp_i \bigcup sp_i \bigcup \{i\}} \left\lceil \frac{B_i^{slow}}{T_j} \right\rceil C_j \tag{2.6}$$

2.3 Summary

We have explored some methods that can be used to ensure that, for each flow exchanged across a studied network, the end-to-end response time does not exceed its corresponding deadline. The model checking, for example, gives the exact worst-case response time but does not fit for large scale industrial networks. The Network Calculus and the Trajectory Approach form good candidates to address the studied problem. They were both applied on a real industrial network such as the AFDX avionic system. The results showed that the Trajectory Approach gives tighter bounds [48]. Table 2.3 lists the advantages and the drawbacks of each method. In addition, Fig 2.12 illustrates the bounds obtained by the different methods.

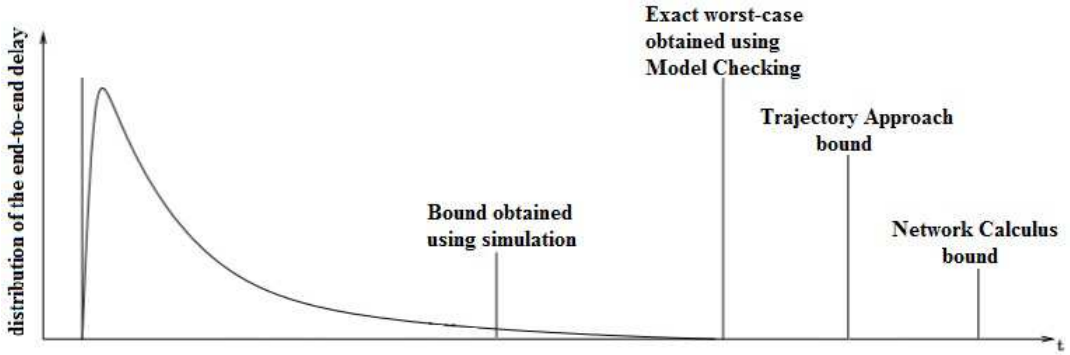


Figure 2.12: End-to-end response time bounds

Method	Advantages	Drawbacks
Model Checking	– exact worst-case response time – provides a counter example	– Completeness of the formalized property. – State space explosion problem
Network Calculus	– deterministic upper bound – Application on lossy systems	– accuracy of the arrival and service curves determines the precision of the upper bound – global service curve is not easy to compute
Trajectory Approach	– can be used on networks holding heterogeneous flows – tighter upper bound (compared to other methods)	– Could only be applied on lossless networks

Table 2.3: Methods used to compute the worst-case end-to-end response time.

2.4 Conclusion

In this chapter, we have briefly explored some methods that can be used to perform timing analysis. In particular, we are asked to determine an upper bound of the end-to-end response time of flows exchanged across the studied network. These methods can be divided into two groups: the simulation-based and the analytical approaches. We recall that, when interested in computing deterministic guarantees, simulation approaches fail to meet the requirements. Among the analytical approaches, we have the model checking, the network calculus and the Trajectory Approach. After comparing these approaches, the trajectory approach presents several advantages. In the next chapter, we investigate the problems of applying this method on large industrial configurations.

Chapter 3

TA limitations for FIFO scheduled flows

Contents

3.1 Introduction

As stated in the previous chapter, the Trajectory Approach is a great candidate to tackle the problem of ensuring the determinism of the deployed network. In this chapter, we are interested in computing the upper bounds of flows scheduled using the FIFO serving policy and having null release jitters. Moreover, all nodes have the same data rate. In section 3.2, the expression of the upper bound as presented by the Trajectory Approach is introduced. An example is given in section 3.3 illustrating the computation process. Then, we discuss the problems confronted when applying this method on large industrial configurations in section 3.4. Then, for simplicity sake and without loss of generality, we illustrate these limitations on small configurations in sections 3.5 and 3.6. Finally, the effect of these limitations is studied in section 3.7 and the conclusion is presented in section 3.8.

3.2 Trajectory Approach for FIFO policy

The FIFO policy is the most used due to its implementation simplicity. This means that there are no flows that have strictly smaller or greater priority level. In this case, lp_i and hp_i are two empty sets. Hence, no delay introduced by these flows is considered. The expression of the latest execution time becomes:

$$\begin{aligned} W_{i,t}^{q} \;\leq\; & (q-1)\cdot L_{max} + \sum_{h=1}^{q-1} \max_{j}\{C_j\} + \left\lfloor \frac{t}{T_i} \right\rfloor C_i \\ & + \sum_{j \neq i} \left(1 + \left\lfloor \frac{t + S_{max_j}^{first_{i,j}} - S_{min_j}^{first_{i,j}} + S_{max_i}^{first_{i,j}} - M_i^{first_{i,j}}}{T_j} \right\rfloor \right) C_j \end{aligned} \tag{3.1}$$

Hence, the expression of $W_{i,t}^q$ can also be written as follows:

$$\begin{aligned} W_{i,t}^q &\leq (q-1) \cdot L_{max} && (3.2)\\ &+ \sum_{h=1}^{q-1} \max_j \{C_j\} && (3.3)\\ &+ \left\lfloor \frac{t}{T_i} \right\rfloor C_i && (3.4)\\ &+ \sum_{j \in sp_i} \left(1 + \left\lfloor \frac{t + A_{i,j}}{T_j} \right\rfloor \right) C_j && (3.5) \end{aligned}$$

with $A_{i,j} = S_{max_j}^{first_{i,j}} - S_{min_j}^{first_{i,j}} + S_{max_i}^{first_{i,j}} - M_i^{first_{i,j}}$.

We recall that, once the expression of the latest execution time is established, computing the upper bound requires testing all time instants until the largest period on the slowest node denoted B_i^{slow} (as defined in Eq. 2.6). The expression of the upper bound is:

$$R_i = \max_{t \leq B_i^{slow}} \{W_{i,t}^q - t + C_i\} \quad (3.6)$$

3.3 Basic computation example

We show the application of the Trajectory Approach on a simple configuration depicted in Fig. 3.1. We consider five flows whose paths are described in Table 3.1. All flows have the same processing time and the same period.

$$\begin{aligned} C_i &= 26\mu s, \forall i \in [1,5] \\ T_i &= 1000\mu s, \forall i \in [1,5] \end{aligned}$$

We recall that the used switches associate to each output port a dedicated buffer. Hence, we say that a flow postpone another one if they share the same output port.

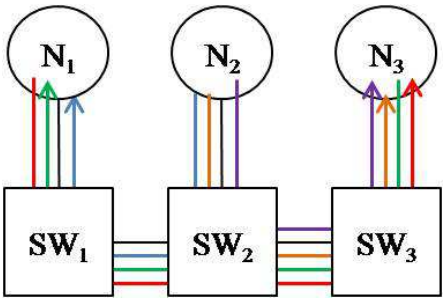


Figure 3.1: Illustrative configuration

flow	Path
τ_1	$\mathcal{P}_1 = \{N_1, SW_1, SW_2, SW_3, N_3\}$
τ_2	$\mathcal{P}_2 = \{N_2, SW_2, SW_3, N_3\}$
τ_3	$\mathcal{P}_3 = \{N_2, SW_2, SW_3, N_3\}$
τ_4	$\mathcal{P}_4 = \{N_2, SW_2, SW_1, N_1\}$
τ_5	$\mathcal{P}_5 = \{N_3, SW_3, SW_2, SW_1, N_1\}$

Table 3.1: Paths of the considered flows

Let us take for example flows τ_1 and τ_5; these flows do not share on any switch the same output port. Therefore, τ_5 does not delay the execution of τ_1 and vice versa. Flows postponing the execution of τ_1 are τ_2 and τ_3 and the first common node in this case is switch SW_2. The latest execution time of a packet belonging to τ_1 and activated at time t can be written as follows:

$$W_{1,t}^{N_3} \leq 4L + 4C + \left\lfloor \frac{t}{T_1} \right\rfloor C_1 + \left(1 + \left\lfloor \frac{t + A_{1,2}}{T_2} \right\rfloor\right) C_2 + \left(1 + \left\lfloor \frac{t + A_{1,3}}{T_3} \right\rfloor\right) C_3$$

where

$$\begin{aligned} A_{1,2} &= S^{SW_2}_{max_1} + S^{SW_2}_{max_2} - S^{SW_2}_{min_2} - M^{SW_2}_1 \\ &= (2C + 2L) + (3C + L) - (C + L) - (2C + 2L) \\ &= 2 \cdot C = 52\mu s. \end{aligned}$$

In the same manner, $A_{1,3} = 2 \cdot C = 52\mu s$.

Hence, the expression of $W^{N_3}_{1,t}$ becomes:

$$W^{N_3}_{1,t} \leq 4 \cdot 3 + 4 \cdot 26 + \left\lfloor \frac{t}{1000} \right\rfloor \cdot 26 + \left(1 + \left\lfloor \frac{t+52}{1000} \right\rfloor\right) \cdot 2 + \left(1 + \left\lfloor \frac{t+52}{1000} \right\rfloor\right) \cdot 26$$

The next step consists of determining the value of the longest busy period B_1. The computation is recursive. We start by $B^0_1 = 1$, then we use the equation introduced in the previous chapter. We stop computation when the series converges.

$$\begin{aligned} B^0_1 &= 1 \\ B^1_1 &= \sum_{\substack{j \in sp_i \bigcup \{i\} \\ \mathcal{P}_i \bigcap \mathcal{P}_j \neq \emptyset}} \left\lceil \frac{B^0_1}{T} \right\rceil \cdot C_j = \left\lceil \frac{1}{T_1} \right\rceil \cdot C_1 + \left\lceil \frac{1}{T_2} \right\rceil \cdot C_2 + \left\lceil \frac{1}{T_3} \right\rceil \cdot C_3 = 3C \\ B^2_1 &= \left\lceil \frac{3C}{T_1} \right\rceil \cdot C_1 + \left\lceil \frac{3C}{T_2} \right\rceil \cdot C_2 + \left\lceil \frac{3C}{T_3} \right\rceil \cdot C_3 = 3C \end{aligned}$$

The upper bound R_1 is expressed below. It is obtained for $t = 0$ and is equal to $194\mu s$.

$$R_1 = \max_{0 \leq t \leq 3C} \{W^{N_3}_{1,t} - t + C\}$$

Table 3.2 depicts the upper bounds of the end-to-end response times of the five flows present in the studied configuration. The results were obtained using a tool which is described in section 3.7 and was developed during this thesis.

flow	τ_1	τ_2	τ_3	τ_4	τ_5
$R_i(\mu s)$	194	191	191	191	168

Table 3.2: Upper bounds obtained using the Trajectory Approach

3.4 Limitations

When applying the Trajectory Approach on a large industrial configuration, we are interested in two main aspects. The first one is the precision of the end-to-end response time upper bound and the second one is the scalability.

1. **Precision**: We know that, for certification purposes, the exchanged flows must respect their deadlines (in other terms, the end-to-end response time should be smaller than the corresponding deadline). If the upper bound is overestimated then it is possible that it might exceed the corresponding deadline, while in reality it is not the case. Thus, to be able to certify the system, it must be overdimensioned leading to higher costs. Consequently, the precision of the computed upper bound is a serious matter.
   We list briefly why the loss of precision of the computed upper bound can occur. In section 3.5, a more detailed analysis is carried out.
   For FIFO policy and as obvious in the expression of $W_{i,t}^q$, the pessimism detected in the method is due to:

   - Overestimating the delay of junction packets which is expressed by Eq. (3.3).
   - Overestimating the delay of packets postponing the studied flow which is expressed by Eq. (3.5).

A measurement of the pessimism was proposed in [49] for an AFDX (Avionic Full DupleX) network.

2. **Scalability**: This parameter is as important as the precision of the upper bound. In fact, the approach must be applied on large industrial configurations that are composed of more than hundred switches and can hold more than thousands of exchanged flows. In this case, the computation can become complex. Yet, the computation time required by this method to determine the upper bound of every flow must be reasonable.

3.5 Precision

The loss of precision in the computed upper bound is mainly due to a precision loss of the expression of $W^q_{i,t}$. As we said before, this can be explained by an overestimation in two terms of $W_{i,t}$ expression.
The first term (i.e. Eq. (3.3)) is the delay introduced by junction packets which are packets counted twice. We recall that a junction packet is a packet that is present at the end of busy period on node h and at the beginning of the busy period on the next node (i.e. node $h+1$).
The second term (i.e. Eq. (3.5)) is in fact the delay introduced by packets postponing the studied flow. In this case, two types of overestimation may occur:

- the first one is because of the summation. In reality, for physical reasons, a flow that shares the same path does not necessarily postpone the considered flow. This is called the serialization effect;

- secondly, the value of $A_{i,j}$ affects the precision of R_i. If overestimated, then the upper bound might be too. This scenario is identified when flows leaving

the path either of the studied flow or another one affecting it.

To illustrate the source of pessimism when using the FIFO serving policy, small configurations are used in the following.

3.5.1 Junction packets

The delay introduced by the junction packets is up to $\sum_{h=1}^{q-1} \max_{j \in hp_i \bigcup sp_i}\{C_j\}$. It is clear that if all flows have the same processing time, then this term does not introduce any pessimism into the upper bound of the end-to-end response time. On the other, if flows do not have the same processing time, the obtained upper bound might no longer be precise. For that, let us consider the following scenario (see Fig. 3.2): the studied flow τ_1 crosses the following node sequence $\mathcal{P}_1 = \{N_2, SW_2, SW_3, N_3\}$. The flow τ_2 has the same source and destination while flow τ_3 is sent from node N_1 to node N_3. Flows τ_1 and τ_3 have the same processing time (i.e. $C_1 = C_3 = 50\mu s$) while the processing time of τ_2 is smaller than that of τ_1 ($C_2 = 25\mu s$). The period of the flows is equal to $2000\mu s$. The delay incurred by the switches is equal to $3\mu s$.

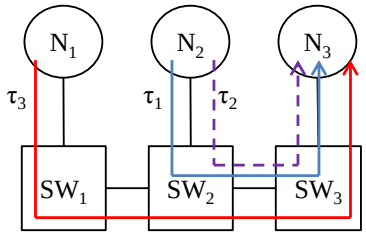


Figure 3.2: Illustrative configuration

The exact worst-case response time is shown in the scheduling diagram (see Fig. 3.3) and is equal to $3L + C_2 + 4C_1 = 234\mu s$. On the other hand, the ex-

pression of the latest execution time $W_{1,t}^{N_3}$ is given in Eq. (3.7).

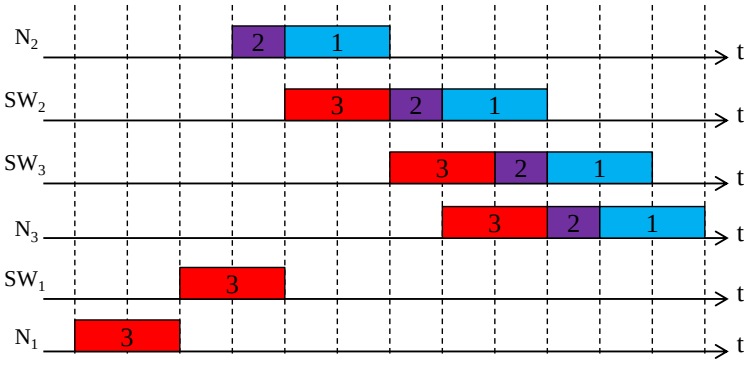


Figure 3.3: Scheduling diagram of flow τ_1

The upper bound of the end-to-end response time can be easily obtained by solving graphically the equation of $R_1 = \max\{W_{1,t}^{N_3} - t + C_1\} = 284\mu s$ (see Fig. 3.4). It is equal to the maximum deviation between $y_1 = W_{1,t}^{N_3} + C_1$ and $y_2 = t$.

$$
\begin{aligned}
W_{1,t}^{q} &\leq 3L + (C_1 + C_1 + C_1) + \left\lfloor \frac{t}{T_1} \right\rfloor C_1 + \left(1 + \left\lfloor \frac{t}{T_2} \right\rfloor\right) C_2 + \left(1 + \left\lfloor \frac{t + C_1}{T_3} \right\rfloor\right) C_3 \\
W_{1,t}^{q} &\leq 3L + 4C_1 + C_2 + \left\lfloor \frac{t}{T_1} \right\rfloor C_1 + \left\lfloor \frac{t}{T_2} \right\rfloor C_2 + \left\lfloor \frac{t + C_1}{T_3} \right\rfloor C_3 \qquad (3.7)
\end{aligned}
$$

3.5.2 Serialization

To understand the serialization effect, let us consider the simple example depicted in Fig. 3.5(a) in which the studied flow τ_1 traverses the sequence $\{N_1, SW_1, SW_2, SW_3, N_3\}$ and two flows (τ_2 and τ_3) follow the same sequence $\{N_2, SW_2, SW_3, N_3\}$. We consider that all flows have the same processing time C. Fig. 3.5(b) shows the exact

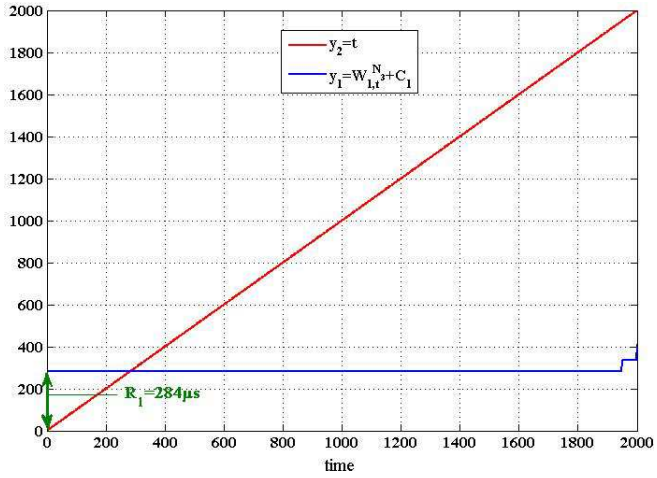
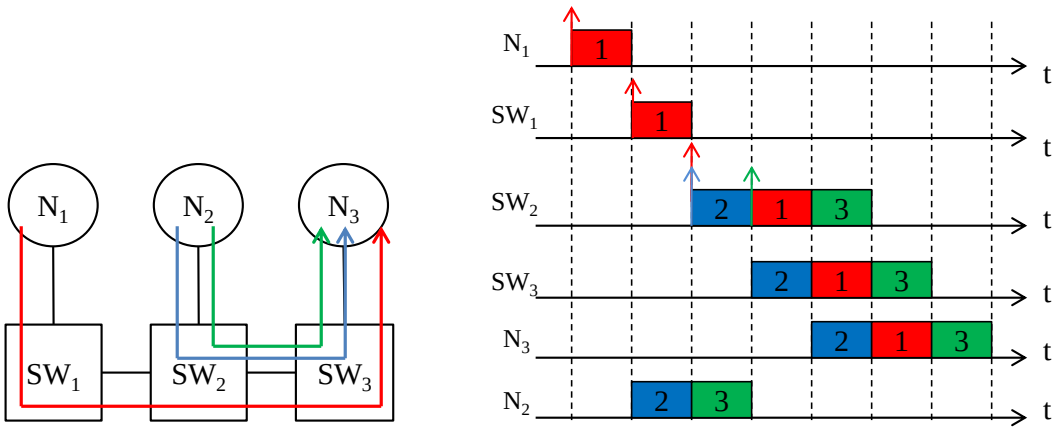


Figure 3.4: graphical solution of $R_1 = \max\{W_{1,t}^{N_3} - t + C_1\}$

worst-case response time of packet p belonging to τ_1. Packets of flow τ_2 and τ_3 are serialized (i.e. processed one after the other) on their source - node N_2. The difference between the arrival time of these packets on node SW_2 is at least equal to the processing time C.

The original calculus considers that both τ_2 and τ_3 postpone the execution of the studied flow. However, in order to postpone the execution of packet p, packets of flow τ_2 and τ_3 should arrive at the same time as the arrival time of p on node SW_2 which is impossible.

In general, the original calculus of the Trajectory Approach considers postponing individual packets without taking into consideration that packets sharing the same links are serialized which means that not all of them postpone the execution of the desired packet. This effect was first presented in [50]. A solution was proposed to reduce it in [48], in which instead of postponing individually each packet, sequence of already serialized packets are postponed.



(a) Illustrative configuration

(b) Scheduling diagram of flow τ_i

Figure 3.5: Serialization example

3.5.3 Effect of leaving flows

Let us consider the following scenario:

1. existence of a flow (or set of flows) leaving the path of τ_i or the path of one of the flows directly or indirectly affecting the response time of the studied flow.

2. existence of a flow meeting τ_i for the first time on a node located after that on which the flow has left.

For instance, let us consider the configuration depicted in Fig. 3.6. The studied flow is τ_1. After flow τ_2 leaves the path of τ_1 on SW_1, another flow (i.e. τ_3) join the path of the studied flow on node SW_3.

If this scenario is present in the studied network, then the Trajectory Approach may introduce an over-estimation into the upper bound.
If the value of $A_{i,j}$ is overestimated and is higher than the period of the inserted

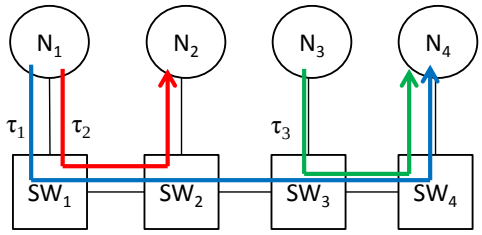


Figure 3.6: Illustrative example

flow, then TA counts additional packets which introduces a pessimism into the upper bound.

We have identified the following scenarios for which the parameter $A_{i,j}$ might be overestimated:

1. flows leaving the path of the studied flow.
2. flows leaving the path of a flow (or set of flows) interacting directly with the studied flow.
3. flows that do not interact directly with the studied flow but leave the path of a flow (or set of flows) interacting indirectly with the studied flow.

In the following, we give simple examples explaining the effect of each of these flows on the precision of the computed upper bound.

3.5.3.1 Flows leaving the path of τ_i

The following configuration (e.g. Fig. 3.7) is used to explain the effect of flows leaving the trajectory of the studied flow. Nine flows coexist in this configuration: the studied flow τ_1 (blue arrow) follows path $\mathcal{P}_1 = \{N_1, SW_1, SW_2, SW_3, SW_4, N_4\}$,

seven flows (green arrow), numbered from τ_2 to τ_8, have $\{N_1, SW_1, SW_2, N_2\}$ as their path and flow τ_9 (red arrow) visits the following sequence $\mathcal{P}_9 = \{N_3, SW_3, SW_4, N_4\}$. In addition, all flows have the same processing time C. τ_9 has a period equal to $4C$ and periods of the other flows are considered to be extremely large.

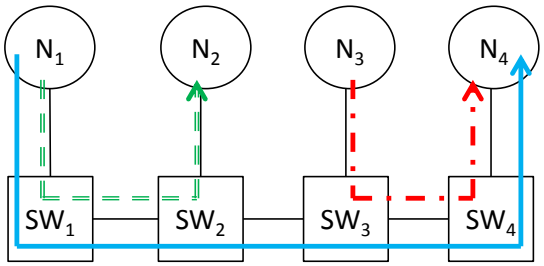


Figure 3.7: Configuration illustrating the effect of flows leaving the path of the studied flow

The latest execution time $W_{1,t}^{N_4}$ of τ_1 becomes:

$$\begin{aligned} W_{1,t}^{N_4} &\leq 5(C+L) + \left\lfloor \frac{t}{T_1} \right\rfloor .C \\ &+ \sum_{j=2}^{8} \left(1 + \left\lfloor \frac{t}{T_j} \right\rfloor\right) C \\ &+ \left(1 + \left\lfloor \frac{t+7C}{T_9} \right\rfloor\right) C \end{aligned}$$

and $R_1 = \max_{0 \leq t \leq B_1} \{W_{1,t}^{N_4} - t + C_1\}$.

Determining B_1 is recursive computation. We start by $B_1^0 = C$ for example, then replace it in the equation of B_1 (defined in the previous chapter) and so on until B_1

converges.

$$
\begin{aligned}
B_1^0 &= C \\
B_1^1 &= \sum_{j=1,2...,8} \left\lceil \frac{B_1^0}{T_j} \right\rceil C_j + \left\lceil \frac{B_1^0}{T_9} \right\rceil C_9 = \sum_{j=1,2...,8} \left\lceil \frac{1}{T} \right\rceil C + \left\lceil \frac{1}{4C} \right\rceil C = 9C \\
B_1^2 &= \sum_{j=1,2...,8} \left\lceil \frac{9C}{T} \right\rceil C + \left\lceil \frac{9C}{4C} \right\rceil C = 11C \\
B_1^3 &= \sum_{j=1,2...,8} \left\lceil \frac{11C}{T} \right\rceil C + \left\lceil \frac{11C}{4C} \right\rceil C = 11C
\end{aligned}
$$

The upper bound of the end-to-end response time of τ_1 is obtained for $t = 0$ and is equal to $15C + 5L$. The exact worst-case response time (EWCRT) of τ_1 is depicted in Fig. 3.8 and is equal to $14C + 5L$.

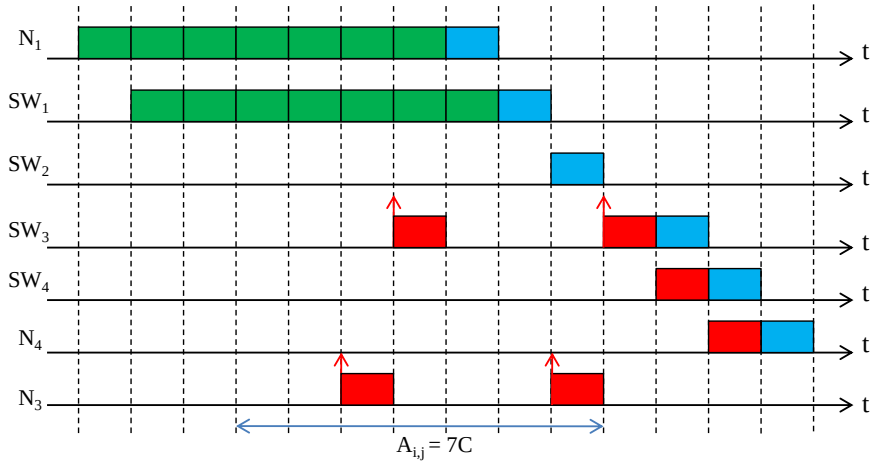


Figure 3.8: worst-case scenario of τ_1

We can notice that the Trajectory Approach has counted two packets of flow τ_9 instead of a single one. This is due to an overestimation of the value of $A_{i,j}$. The

Trajectory Approach considers that all packets activated on node N_3 within an interval of length $A_{i,j} = 7C$ postpone the execution of the studied flow. While in reality, the length of the interval is suppose to be equal to C. We show in section 3.7 that the error introduced by the Trajectory Approach can be worse and depends on the number of leaving flows.

3.5.3.2 Flows leaving the path of one (or more) flow(s) interacting directly with τ_i

Configuration represented in Fig. 3.9 shows the impact of flows leaving the trajectory of a flow that directly interact with the studied flow. In this configuration, we consider nine flows with the following characteristics: the studied flow τ_1 (represented by a blue arrow) follows path $\mathcal{P}_1 = \{N_5, SW_5, SW_6, N_6\}$, seven flows (numbered from τ_2 to τ_8) represented by a red arrow have $\{N_1, SW_1, SW_2, N_2\}$ as their path and flow τ_9 represented by a green arrow visits the following sequence $\{N_1, SW_1, SW_2, SW_3, SW_4, SW_5, SW_6, N_6\}$. We consider that all flows have the same processing time C. In addition, τ_9 has a period equal to $4C$ and periods of the other flows are considered to be extremely large compared to C.

The latest execution time of τ_1 on its last visited node becomes:

$$W_{1,t}^{N_6} \leq 3(C + L) + \left\lfloor \frac{t}{T_1} \right\rfloor .C + \left(1 + \left\lfloor \frac{t + 7C}{T_9} \right\rfloor\right) C$$

and $R_1 = \max_{0 \leq t \leq 2C}\{W_{1,t}^{N_6} - t + C_1\}$ The upper bound of the end-to-end response time is obtained for $t = 0$ and is equal to $6C + 3L$. Flows going from node N_1 to N_2 do not interact directly with the studied flow, yet their effect is present in the value of $A_{1,9}$. The exact worst case response time (EWCRT) is equal to $5C + 3L$. Once

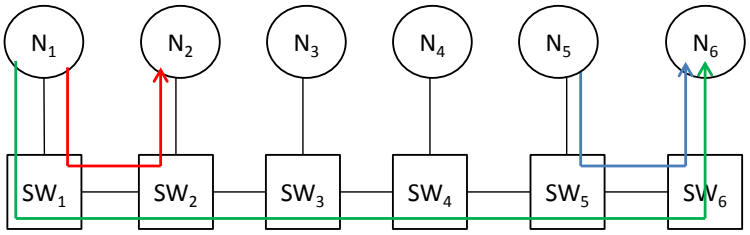


Figure 3.9: Configuration illustrating the effect of flows directly interacting with the studied flow.

again, the Trajectory Approach has counted a single packet in excess. Similarly, the value of $A_{1,9}$ is the reason of the error introduced by the Trajectory Approach.

3.5.3.3 Flows leaving the path of one (or more) flow(s) interacting indirectly with τ_i

We consider the following configuration (depicted in Fig. 3.10). The studied flow τ_1 traverses the node sequence $\mathcal{P}_1 = \{N_6, SW_6, SW_7, SW_8, N_8\}$ and is represented by a blue arrow. Flow τ_2 follows the node sequence $\mathcal{P}_2 = \{N_4, SW_4, SW_5, SW_6, SW_7, N_7\}$ and its period is $2C$. Flow τ_3, represented by a green arrow, follows the sequence $\mathcal{P}_3 = \{N_1, SW_1, SW_2, SW_3, SW_4, SW_5, N_5\}$ and its period is $4C$. In addition, four flows rejoin on node SW_2 the path of τ_3 and leave it on node SW_3. They are represented by a purple arrow. The periods of the other flows are extremely large.

The latest execution time of τ_1 on node N_8 is given by the expression (3.8).

$$W_{1,t}^{N_8} \leq 4(C+L) + \left\lfloor \frac{t}{T_1} \right\rfloor .C + \left(1 + \left\lfloor \frac{t + A_{1,2}}{T_2} \right\rfloor \right) C \tag{3.8}$$

where $A_{1,2} = S_{max_1}^{SW_6} + S_{max_2}^{SW_6} - M_1^{SW_6} - S_{min_2}^{SW_6}$.

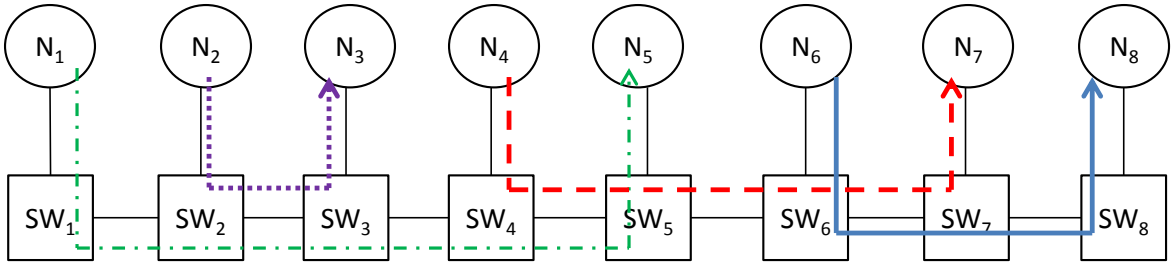


Figure 3.10: Configuration illustrating the effect of flows indirectly interacting with the studied flow

We have $S_{max_1}^{SW_6} - M_1^{SW_6} = 0$ and $S_{min_2}^{SW_6} = 3(C + L)$.
Determining $S_{max_2}^{SW_6}$ requires computing $R_2^{SW_5}$.

The latest execution time of τ_2 on node SW_5 is given by expression (3.9).

$$W_{2,t}^{SW_5} \leq 2(C + L) + \left\lfloor \frac{t}{T_2} \right\rfloor .C + \left(1 + \left\lfloor \frac{t + 4C}{T_3} \right\rfloor\right) C \tag{3.9}$$

The worst-case response time of τ_2 on node SW_5 is obtained at $t = 0$ and is upper bounded by $R_2^{SW_5} = 5C + 2L$. The value of $S_{max_2}^{SW_6}$ becomes equal to $5C + 3L$. After replacing $S_{max_2}^{SW_6}$ by its value in Eq. (3.8), the expression of $W_{1,t}^{N_8}$ becomes:

$$W_{1,t}^{N_8} \leq 4(C + L) + \left\lfloor \frac{t}{T_1} \right\rfloor .C + \left(1 + \left\lfloor \frac{t + 2C}{T_2} \right\rfloor\right) C \tag{3.10}$$

The worst-case response time of τ_1 is obtained for $t = 0$ and is upper bounded by $R_1^{N_8} = 7C + 4L$, while the exact worst case response time has a value of $6C + 4L$. Although flows activated on N_2 do not interact directly with τ_1, yet they have affected the precision of the upper bound R_1.

3.6 Scalability

Even if a method offers the exact worst-case response time, yet it is vital that it gives results in an acceptable time frame when applied on large industrial configurations. We show on a simple example, the reason for which computing the upper bound of a flow τ_i can be complex.

Let us consider the configuration represented in Fig 3.11. The studied flow τ_5 is processed on the node sequence $P_5 = \{N_5, SW_5, SW_6, N_6\}$.

All flows have the same processing time C and the same period T. The expression of the upper bound R_5 on node N_6 is given in Eq. (3.11).

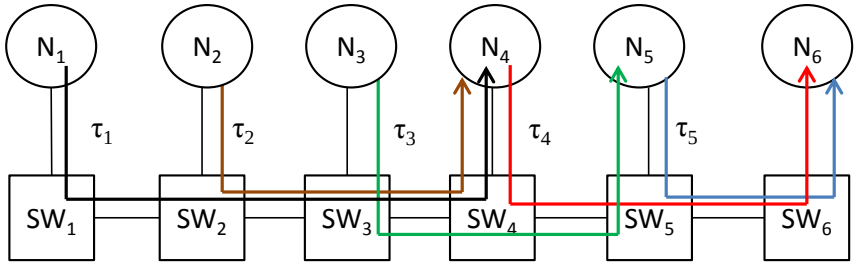


Figure 3.11: Representative example

$$R_5^{N_6} = \max_{t \geq 0} \left\{ W_{5,t}^{N_6} - t + C \right\} \tag{3.11}$$

and

$$\begin{aligned} W_{5,t}^{N_6} &= 3(L + C) + \left(1 + \left\lfloor \frac{t}{T} \right\rfloor\right) \cdot C \\ &+ \left(1 + \left\lfloor \frac{t + A_{4,5}}{T} \right\rfloor\right) \cdot C - C \end{aligned} \tag{3.12}$$

To compute $W_{5,t}$, we need to determine the value of $A_{4,5}$ which is the sum of the jitters of τ_4 and τ_5 on node SW_5.

$$A_{4,5} = \left(S_{max_4}^{SW_5} - M_4^{SW_5}\right) + \left(S_{max_5}^{SW_5} - S_{min_5}^{SW_5}\right)$$

Since all flows have the same processing time then $M_4^{SW_5} = S_{min_4}^{SW_5}$ where $S_{min_j}^h$ (respectively $S_{max_j}^h$) is the minimum (respectively the maximum) time required by a packet of τ_j to reach node h.

Computing $S_{max_4}^{SW_5}$ requires determining its response time on the previous node (e.g node SW_4). Similarly, determining the response time of τ_4 on SW_4 requires calculating the value of the response time of τ_3 on node SW_3 and so on. This makes the computation process recursive.

Moreover, we need to test several instants t to derive the worst-case upper bound. If flows have large periods then the set of instants to test is large.

3.7 Numerical evaluation on sample configurations

In this section, we first increase the number of serialized flows and evaluate their impact on the upper bound provided by the Trajectory Approach. At the next step, the impact of leaving flows on the upper bound offered by the Trajectory Approach is observed. For this purpose, a Trajectory Approach tool was developed. The program takes a text file (.txt) in its input. Each line of the file contains information about the exchanged flows such as their processing time, their period and the set of nodes crossed by each flow. The output file contains the end-to-end response time's upper bound of each flow and the time required to compute it.

The configurations presented in section 3.5 are used. Small configurations were chosen allowing us to compute the exact worst-case response time. For this purpose, a tool was also developed; it exhaustively checks all possible flows activation searching

for the worst-case scenario experienced by a flow.

We consider that all flows have the same processing time on all nodes and is equal to $26\mu s$. The switching delay L is equal to $3\mu s$.

3.7.1 Impact of serialization

The configuration under study is depicted in Fig. 3.5(a); it consists of six nodes. The flow under study is τ_1, it follows the sequence $\{N_1, SW_1, SW_2, SW_3, N_3\}$. Several flows are being serialized before joining the path of flow τ_1. These flows follow the sequence $\{N_2, SW_2, SW_3, N_3\}$. Periods of flows are extremely large.

We increase the number of serialized flows and observe their impact on the upper bound computed using the Trajectory Approach. The exact worst-case response time is equal to $168\mu s$. Fig. 3.12 shows that the upper bound calculated by the Trajectory Approach becomes pessimistic as the number of serialized flow increases.

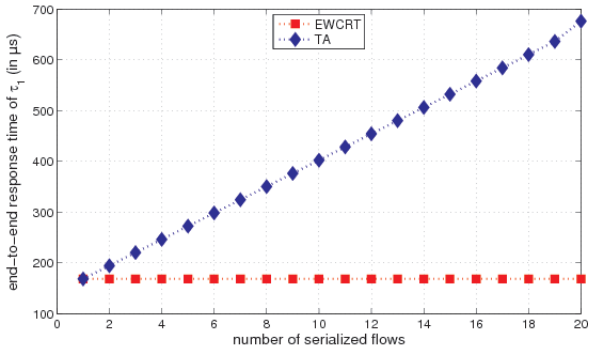


Figure 3.12: Impact of serialization on the Trajectory approach's upper bound

3.7.2 Impact of leaving flows

To evaluate the impact of leaving flows that affect directly or indirectly the response time of the studied flow, we increase on small configurations the number of these flows and observe their influence on the Trajectory Approach's upper bound.

3.7.2.1 Flows leaving the path of the studied flow τ_i

The configuration represented in Fig. 3.7 is used to study the effect of increasing the number of flows interacting directly with the studied flow. Flows in this configuration have the same characteristics as listed previously. The flow under study is τ_1 and crosses the sequence $\{N_1, SW_1, SW_2, SW_3, SW_4, N_4\}$. An additional flow τ_2 is inserted into the path of τ_1 and has a path composed of the sequence $\{N_3, SW_3, SW_4, N_4\}$. A set of flows τ_n $(n = 3, 4, ...)$ traverses nodes $\{N_1, SW_1, SW_2, N_2\}$; these flows leave the path of τ_1 on node SW_2. Fig. 3.13 shows how this kind of flows affect the tightness of the Trajectory Approach upper bound.

The latest starting time of flow τ_1 is:

$$\begin{aligned} W^7_{1,t} &\leq 5(C+L) + \left\lfloor \frac{t}{T_1} \right\rfloor .C + \sum_{j=2}^{8} \left(1 + \left\lfloor \frac{t}{T_j} \right\rfloor\right) C \\ &+ \left(1 + \left\lfloor \frac{t + C.n_{LF}}{T_9} \right\rfloor\right) C \end{aligned}$$

n_{LF} corresponds to the number of leaving flows.

The fourth parameter in the latter equation increases when $C \cdot n_{LF}$ is a multiple of $T_9 = 4C$. When n_{LF} becomes a multiple of 4, the Trajectory Approach's upper bound becomes more pessimistic. This fits with results of Fig. 3.13 in which the error introduced by the Trajectory Approach becomes more important in the interval

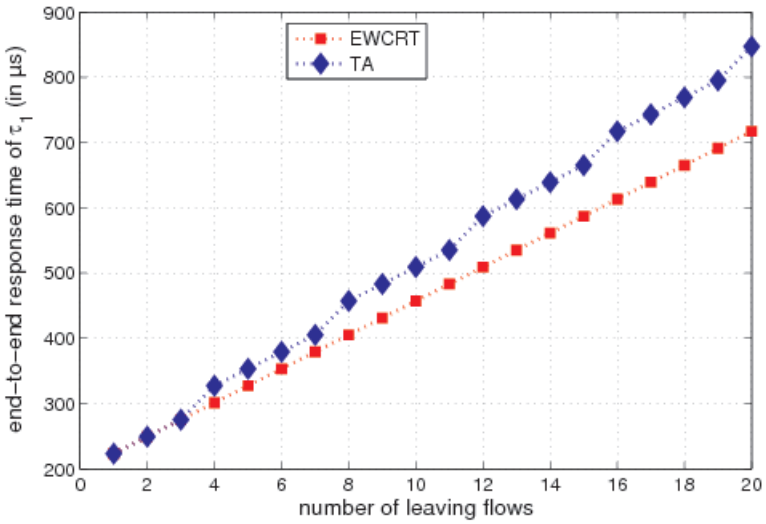


Figure 3.13: Impact of leaving flows on the Trajectory Approach upper bound

[8, 11] than in [4, 7].

3.7.2.2 Flows leaving the path of a flow interacting directly with τ_i

The configuration used to evaluate the effect of these flows on the upper bound is depicted in Fig. 3.9. The studied flow τ_1 follows the sequence $\{N_5, SW_5, SW_6, N_6\}$. An additional flow exists in the considered configuration and has N_1 as source and N_6 as destination. A set of flows are sent from node N_1 to node N_2. We increase the number of these flows and observe their impact on the Trajectory Approach upper bound. In Fig. 3.14, the EWCRT and the Trajectory Approach's upper bound are shown. The upper bound computed by the Trajectory Approach increases by step. In this case, each time the number of leaving flows becomes a multiple of $4C$, the Trajectory Approach's upper bound becomes more pessimistic.

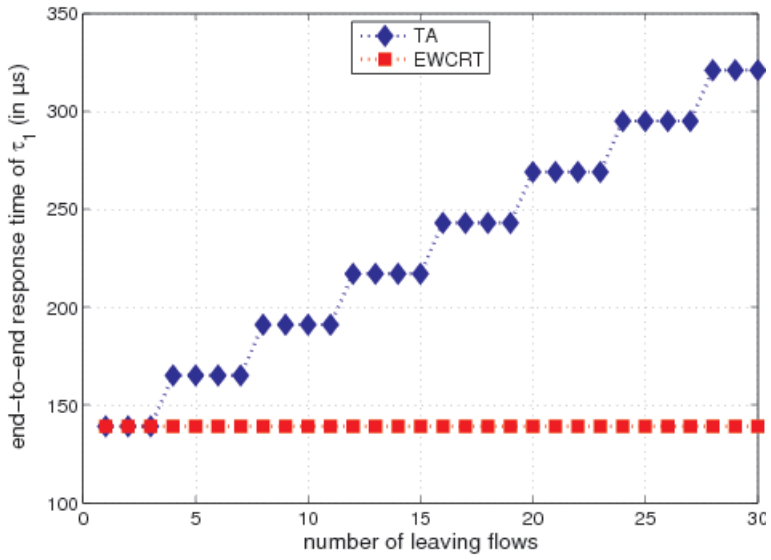


Figure 3.14: Impact of leaving flows on the upper bound of τ_1

3.7.2.3 Flows leaving the path of a flow interacting indirectly with τ_i

The configuration used to evaluate the effect of these flows on the upper bound is depicted in Fig. 3.10. We are interested in flow τ_1 that traverses the node sequence $\mathcal{P}_1 = \{N_6, SW_6, SW_7, SW_8, N_8\}$. Flow τ_2 follows the node sequence $\mathcal{P}_2 = \{N_4, SW_4, SW_5, SW_6, SW_7, N_7\}$ and its period is $2C$. Flow τ_3 follows the sequence $\mathcal{P}_3 = \{N_1, SW_1, SW_2, SW_3, SW_4, SW_5, N_5\}$ and its period is $4C$. In addition, a set of flows τ_n rejoin on node SW_2 the path of τ_3 and leave it on node SW_3. We increase the number of these flows and compute the end-to-end response time of τ_1 using the Trajectory Approach.

Fig. 3.15 shows both the EWCRT of τ_1 and the Trajectory Approach's upper bound. As the number of flows of set τ_n increases, TA upper bound loses its precision.

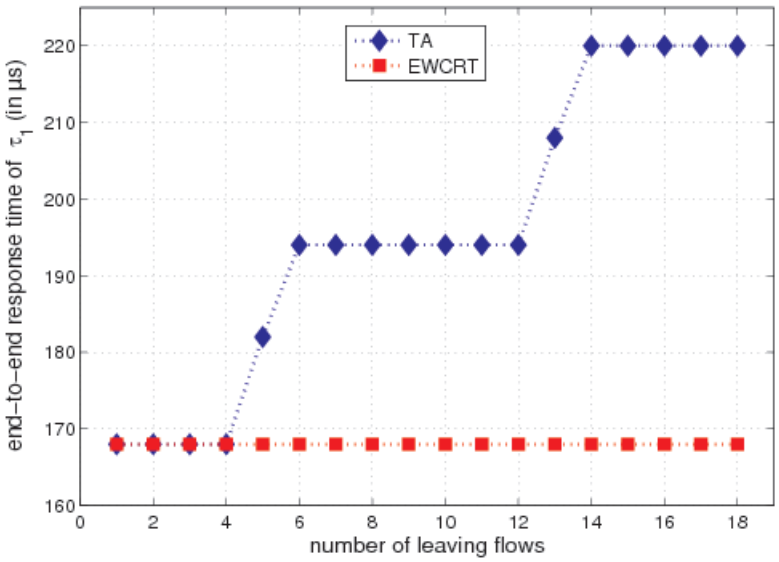


Figure 3.15: Impact of leaving flows on the upper bound of τ_1

3.7.3 Scalability

In the following, we study the ability of the approach of offering results in an acceptable time frame. We start first by comparing the total runtime required by the Trajectory Approach (TA) and the Enhanced Trajectory Approach (ETA). Then we show the impact of increasing both the number of flows and the number of switches on the total runtime.

3.7.3.1 Comparison between TA and ETA

In this analysis, we consider that all flows have the same processing time ($C = 26\mu s$) and the same period ($T = 100ms$). We assume that the switching delay is equal to $3\mu s$. Moreover, the sources and the destinations are chosen randomly. We increase

the number of flows and compare the total runtime required by TA and ETA. Results are depicted in Fig. 3.16.

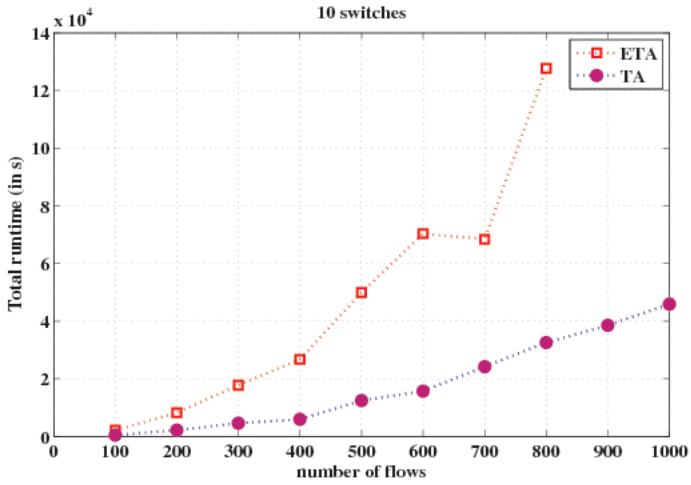


Figure 3.16: Total runtime required by TA and ETA

The number of flows varies from 100 to 1000. As the number of flows increase, both TA and ETA requires more time to compute the upper bounds of all flows. However, as the number of flows increase, ETA becomes far more slower than TA.

3.7.3.2 TA Average runtime

We investigate the impact of both increasing the number of switches and the number of flows on the total runtime required to compute the upper bounds of all flows present in the corresponding configuration.

The number of flows varies from 100 to 1000 while the number of switches varies from 10 to 100. For a fix number of flows and switches, 50 configurations are generated

and for each configuration, the sources and the destinations are randomly chosen. We consider that the packet processing times are equal to $1\mu s$ and the periods are equal to $1000\mu s$. We assume that the switching delay is considered to be null. Fig. 3.17 shows the average total runtime. As the number of switches and number of flows increase, the average runtime becomes more important.

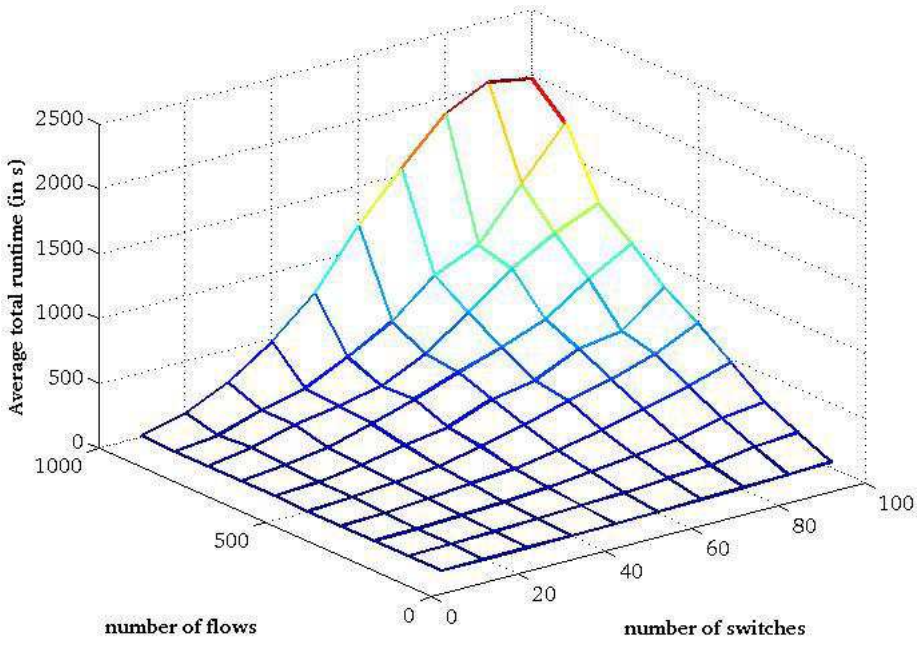


Figure 3.17: Average runtime in function of the number of switches and flows

3.8 Conclusion

In this chapter, we have investigated the limitations of applying the Trajectory Approach for FIFO scheduled flows. The first one is the precision of the upper bound. We have identified the flow's configurations that introduce pessimism into the com-

puted upper bound. The second one is the scalability of the approach. We have shown that the Enhanced Trajectory Approach requires more time than the Trajectory Approach to analyze the same configuration. As much as the precision of the upper bound is important, the scalability of the method is too. For that, in the next chapter, we compute an upper bound of the end-to-end response time but in a reduced time frame.

Chapter 4

Scalable Trajectory Approach

Contents

4.1 Introduction

Since we apply the Trajectory Approach on large industrial configuration in which the amount of interacting flows can be huge, the approach as originally developed fails in offering results in an acceptable time frame. As stated in the previous chapter, the main reasons for which computing the upper bound can be complex are the recursive and the iterative process. We are interested in developing a scalable version of the Trajectory Approach that offers a trade off between the precision of the upper bound and the total time required to calculate the upper bound of each flow in the studied network. In this chapter, a solution allowing to get rid of the recursive process is first introduced in section 4.2. Then, we prove in section 4.2.2 that, for flows verifying specific conditions, the set of instants to be tested can be reduced to a single time instant. In section 4.2.3, the case for which all flows have the same processing time and the same period is discussed. Finally, in section 4.3, we compare results obtained using our proposition and the Enhanced Trajectory Approach and we show that our proposition allows obtaining results on large industrial networks in an acceptable time frame. A conclusion is presented in section 4.4.

4.2 Scalable Trajectory Approach

The recursive process is inducted by flows delaying the studied flow on its first crossed switch. To reduce the computational runtime, we propose at a first step to simplify the expression of the upper bound established using the Trajectory Approach. This can be achieved by eliminating the terms incurring the recursive process.
We consider a sporadic flow τ_i and SW_{first_i} is the first switch crossed by the studied flow τ_i. Let τ_j be a flow activated on the source of the studied flow and τ_k a flow

meeting the studied one for the first time on SW_{first_i}. Flow postponing τ_i on a node other than the source and SW_{first_i} is denoted τ_k. $\tau_j, \tau_k,$ and τ_m are depicted in Fig. 4.1.

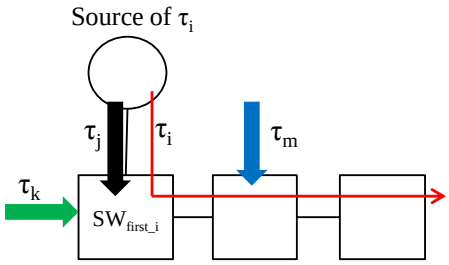


Figure 4.1: Notation illustration

The upper bound R_i as defined by the Enhanced Trajectory Approach (ETA) can be written as follows:

$$R_i = (|P_i| - 1)\, L + \sum_{\substack{h \in P_i \\ h \neq last_i}} \left(\max_{\substack{\{j\} \\ h \in P_j}} \{C_j\} \right) \tag{4.1}$$

$$+ \max_{t \geq 0} \left\{ \sum_{\substack{i \bigcup j \\ first_j = first_i}} \left(1 + \left\lfloor \frac{t}{T_j} \right\rfloor \right) \cdot C_j \right. \tag{4.2}$$

$$+ \sum_{\substack{k \\ first_{i,k} = SW_{first_i}}} \left(1 + \left\lfloor \frac{t + A_{i,k}}{T_k} \right\rfloor \right) \cdot C_k \tag{4.3}$$

$$+ \sum_{\substack{m \\ first_m \neq first_i \\ first_{i,m} \neq SW_{first_i}}} \left(1 + \left\lfloor \frac{t + A_{i,m}}{T_m} \right\rfloor \right) \cdot C_m \tag{4.4}$$

$$\left. - \Delta_i^{SW_{first_i}} - \sum_{\substack{h \\ h \neq \{first_i, SW_{first_i}\}}} \Delta_i^h - t \right\} \tag{4.5}$$

Term (4.2) is the delay incurred by flows activated on the source (denoted $first_i$) of the studied flow.

Term (4.3) is the delay of flows postponing the studied flow on its first crossed switch denoted SW_{first_i}.

Term (4.4) represents the delay introduced by flows postponing τ_i on nodes other than its source and the first switch.

where Δ_i^h is the maximum between zero and $l_x^h - l_0^h$ on node h, since there is only one other input other than IP_0^h (the input of the studied flow on the corresponding switch). We recall that $l_x^{SW_{first_i}}$ is the delay incurred by flows postponing the studied flow on the first switch. $l_0^{SW_{first_i}}$ is the delay of flows competing with the studied flow over the resources on the output port of nodes $first_i$ and SW_{first_i}. The expressions of l_x^h and l_0^h, for $h = SW_{first_i}$, are given respectively in Eq. (4.6) and Eq. (4.7).

$$l_x^{SW_{first_i}} = \text{Eq.(4.3)} - \max_{IP_x^{SW_{first_i}}} \{C_k\} \tag{4.6}$$

$$l_0^{SW_{first_i}} = \sum_{\substack{i \bigcup j \\ first_j = first_i \\ last_{i,j} \neq SW_{first_i}}} \left(1 + \left\lfloor \frac{t}{T_j} \right\rfloor\right) \cdot C_j - \min_{IP_0^{SW_{first_i}}} \{C_j\} \tag{4.7}$$

In fact, the recursive process is introduced by flows delaying the studied flow on the first switch. It is presented by the term (4.3). The idea consists of eliminating this term (4.3). Since it reappears in Δ on the first crossed switch, we have to study the cases for which the value of Delta on this switch is positive and is null.

If Delta is positive this means that Eq. (4.6) is greater than Eq. (4.7). By replacing the value of Delta in Eq. (4.5), we eliminate the effect of flows creating the recusive process. In fact, Eq. (4.3) can be replaced by l_0 plus the maximum process-

ing time of a packet that meet the studied flow for the first time on SW_{first_i}.

If Delta on SW_{first_i} is null then l_x is smaller than l_0 which means that Eq. (4.3) is smaller than l_0 on this switch plus the maximum processing time of a packet that meet the studied flow for the first time on SW_{first_i}.

Now, by considering that the sum of Deltas in Eq. (4.5) is null, an upper bound of R_i is obtained (see Eq. (4.8)). In fact, such hypothesis allows us to determine an upper bound of R_i without the need of computing the value of $A_{i,k}$ of flows postponing τ_i on the first switch SW_{first_i}. Hence, the runtime required to determine the upper bound of a flow is reduced.

Finally, the expression of R_i becomes:

$$
\begin{aligned}
R_i \leq (|P_i| - 1)\, L + \sum_{\substack{h \in P_i \\ h \neq last_i}} \left(\max_{\substack{j \\ h \in P_j}} \{C_j\} \right) + \max_{h = SW_{first_i}} \{C_k\} - \min_{h = SW_{first_i}} \{C_j\} \\
+ 2 \cdot \sum_{\substack{i \bigcup j \\ first_j = first_i \\ last_{i,j} \neq SW_{first_i}}} C_j + \sum_{\substack{j \\ first_j = first_i \\ last_{i,j} = SW_{first_i}}} C_j + \sum_{\substack{m \\ first_m \neq first_i \\ first_{i,m} \neq SW_{first_i}}} C_m \\
+ \max_{t \geq 0} \left\{ 2 \cdot \sum_{\substack{i \bigcup j \\ first_j = first_i \\ last_{i,j} \neq SW_{first_i}}} \left\lfloor \frac{t}{T_j} \right\rfloor \cdot C_j + \sum_{\substack{j \\ first_j = first_i \\ last_{i,j} = SW_{first_i}}} \left\lfloor \frac{t}{T_j} \right\rfloor \cdot C_j \right. \\
\left. + \sum_{\substack{m \\ first_m \neq first_i \\ first_{i,m} \neq SW_{first_i}}} \left\lfloor \frac{t + A_{i,m}}{T_m} \right\rfloor \cdot C_m - t \right\}
\end{aligned}
\tag{4.8}
$$

We recall that $A_{i,m} = \left(S_{max_i}^{SW_{first_i}} - M_i^{SW_{first_i}} \right) + \left(S_{max_m}^{SW_{first_i}} - S_{min_m}^{SW_{first_i}} \right)$. It is computed on the first common node between the paths of flows τ_m and τ_i.

The flow τ_m postpone the execution of the studied flow on any node except its source and the first switch. Hence, the source of τ_m when the first common node is not the last switch crossed by τ_i is at one hop of the common node. In order to compute S_{max_m}, we simply need to determine the flow response time on its source $+L$. Computing S_{max_i} requires determining the response time of τ_i on the previous node which in its turn depends on its response time on the first switch SW_{first_i}. Based on Eq. (4.8), R_i on SW_{first_i} can be computed without the need of determining the value of $A_{i,k}$ of flows postponing the studied flow on this node. Hence, computing R_i requires determining less parameters.

4.2.1 Basic computation example

Let us consider the example illustrated in the Figure. The characteristics of the flow exchanged across the network are presented in Table 4.1. We show the steps used to compute an upper bound of the end-to-end response time of flow τ_1. Let us assume that the switching delay L is null.

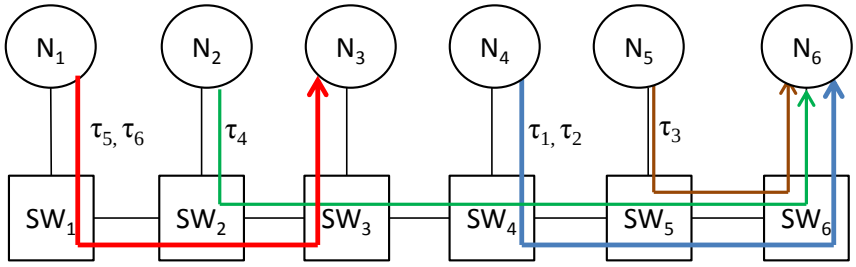


Figure 4.2: Illustrative example

Let us first determine the value of $A_{1,3}$ on the first common node between these flows and which is SW_3. $A_{1,3}$ is equal to the sum of the jitters of τ_1 and τ_3. The

flow	C	T	Path
1	10	100	$\{N_4, SW_4, SW_5, SW_6, N_6\}$
2	10	100	$\{N_4, SW_4, SW_5, SW_6, N_6\}$
3	20	50	$\{N_5, SW_5, SW_6, N_6\}$
4	40	200	$\{N_2, SW_2, SW_3, SW_4, SW_5, SW_6, N_6\}$
5	10	200	$\{N_1, SW_1, SW_2, SW_3, N_3\}$
6	10	200	$\{N_1, SW_1, SW_2, SW_3, N_3\}$

Table 4.1: Characteristic of the flows

jitter of τ_3 is equal to null because the minimum and the maximum arrival time of τ_3 on SW_3 are identical ($S_{max_3} = S_{min_3} = C_3 + L$).

The minimum arrival time of τ_1 denoted S_{min_1} on node SW_1 is equal to $2(C_1 + L)$, while the maximum arrival time S_{max_1} is equal to $R_1^{S_3} + L$. The expression of R_1 on node SW_3 is given as follows:

$$\begin{aligned} R_1^{SW_3} &\leq (C_1 + C_4 + C_4 + C_4) + (C_4 - C_1) + 2(C_1 + C_2) \\ &\quad + \max_{t \geq 0} \left\{ 2 \left(\left\lfloor \frac{t}{T_1} \right\rfloor C_1 + \left\lfloor \frac{t}{T_2} \right\rfloor C_2 \right) - t \right\} \\ R_1^{SW_3} &\leq 4C_4 + 2(C_1 + C_2) + \max_{t \geq 0} \left\{ 4 \left\lfloor \frac{t}{100} \right\rfloor C_1 - t \right\} \end{aligned}$$

The maximum is obtained for $t = 0$, $R_1^{SW_3}$ is equal to 170 and $A_{1,3} = 150$.

Using Eq. (4.8), the upper bound of the end-to-end response time of τ_1 is equal to:

$$R_1 \leq 4C_4 + 2(C_1 + C_2) + \max_{t \geq 0} \left\{ 2 \left(\left\lfloor \frac{t}{100} \right\rfloor C_1 + \left\lfloor \frac{t}{100} \right\rfloor C_2 \right) + \left\lfloor \frac{t + 150}{50} \right\rfloor C_3 - t \right\}$$

As defined in [42], the set of instants to test is such that $t \leq B_1^{slow}$.
B_1^{slow} is the largest busy period and is defined by Eq. (4.9).

$$B_i^{slow} = \sum_{\substack{i \bigcup j \\ P_i \bigcap P_j \neq \emptyset}} \left\lceil \frac{B_1^{slow}}{T_j} \right\rceil C_j \tag{4.9}$$

$$B_i^{slow} = \left\lceil \frac{B_1^{slow}}{T_1} \right\rceil C_1 + \left\lceil \frac{B_1^{slow}}{T_2} \right\rceil C_2 + \left\lceil \frac{B_1^{slow}}{T_3} \right\rceil C_3 + \left\lceil \frac{B_1^{slow}}{T_4} \right\rceil C_4$$

The next step consists of determining the value of B_i^{slow} which is a numerical sequence that converge towards a constant value. Let us start with $B_i^0 = 1$.

$$\begin{aligned}
B_1^1 &= 2 \left\lceil \frac{B_1^0}{100} \right\rceil C_1 + \left\lceil \frac{B_1^0}{50} \right\rceil C_3 + \left\lceil \frac{B_1^0}{200} \right\rceil C_4 \\
&= 2 \left\lceil \frac{1}{100} \right\rceil C_1 + \left\lceil \frac{1}{50} \right\rceil C_3 + \left\lceil \frac{1}{200} \right\rceil C_4 = 2C_1 + C_3 + C_4 = 80 \\
B_1^2 &= 2 \left\lceil \frac{B_1^1}{100} \right\rceil C_1 + \left\lceil \frac{B_1^1}{50} \right\rceil C_3 + \left\lceil \frac{B_1^1}{200} \right\rceil C_4 \\
&= 2 \left\lceil \frac{80}{100} \right\rceil C_1 + \left\lceil \frac{80}{50} \right\rceil C_3 + \left\lceil \frac{80}{200} \right\rceil C_4 = 2C_1 + 2C_3 + C_4 = 100 \\
B_1^3 &= 2 \left\lceil \frac{100}{100} \right\rceil C_1 + \left\lceil \frac{100}{50} \right\rceil C_3 + \left\lceil \frac{100}{200} \right\rceil C_4 = 2C_1 + 2C_3 + C_4 = 100
\end{aligned}$$

Hence, B_1^{slow} is equal to 100. Computing the upper bound requires testing every instant between 0 and B_1^{slow}. In this case, R_1 is obtained for $t = 0$ and is equal to 200

4.2.2 Reducing the set of instants to test

In the following, we prove that for flows verifying a specific condition determining the upper bound of the end-to-end response time does not require testing all time

instants. It suffices to check the instant $t = 0$. In other words, the maximum of the Eq.(4.8) is supposed to be reached for the instant $t = 0$.

Proof. Let $f(t)$ be the function declared below. We need to determine its maximum.

$$f(t) = 2 \cdot \sum_{\substack{i \bigcup j \\ first_j = first_i \\ last_{i,j} \neq SW_{first_i}}} \left\lfloor \frac{t}{T_j} \right\rfloor C_j + \sum_{\substack{j \\ first_j = first_i \\ last_{i,j} = SW_{first_i}}} \left\lfloor \frac{t}{T_j} \right\rfloor C_j + \sum_{\substack{m \\ first_m \neq first_i \\ first_{i,m} \neq SW_{first_i}}} \left\lfloor \frac{t + A_{i,m}}{T_m} \right\rfloor C_m - t$$

We note T_{min} (respectively T_{max}) the minimum (respectively maximum) period among flows postponing the studied flow.
We have $f(t) \leq g(t)$ for every instant t such that $g(t)$ is giving as below.

$$g(t) = 2 \sum_{\substack{i \bigcup j \\ first_j = first_i \\ last_{i,j} \neq SW_{first_i}}} \left\lfloor \frac{t}{T_{min}} \right\rfloor C_j + \sum_{\substack{j \\ first_j = first_i \\ last_{i,j} = SW_{first_i}}} \left\lfloor \frac{t}{T_{min}} \right\rfloor C_j + \sum_{\substack{m \\ first_m \neq first_i \\ first_{i,m} \neq SW_{first_i}}} \left\lfloor \frac{t + A_{i,m}}{T_{min}} \right\rfloor C_m - t$$

Next, we determine the maximum of $g(t)$. For that, we can test only the instants $t = 0, KT_{min},$ and $KT_{min} - A_{i,m}$ with $K \in \mathbb{N}^*$

– The expression of $g(KT_{min})$ is giving below.

$$g(KT_{min}) = 2 \sum_{\substack{i \bigcup j \\ first_j = first_i \\ last_{i,j} \neq SW_{first_i}}} \left\lfloor \frac{KT_{min}}{T_{min}} \right\rfloor C_j + \sum_{\substack{j \\ first_j = first_i \\ last_{i,j} = SW_{first_i}}} \left\lfloor \frac{KT_{min}}{T_{min}} \right\rfloor C_j + \sum_{\substack{m \\ first_m \neq first_i \\ first_{i,m} \neq SW_{first_i}}} \left\lfloor \frac{KT_{min} + A_{i,m}}{T_{min}} \right\rfloor C_m - KT_{min}$$

$$g(KT_{min}) = K \left(2 \sum_{\substack{i \bigcup j \\ first_j = first_i \\ last_{i,j} \neq SW_{first_i}}} C_j + \sum_{\substack{j \\ first_j = first_i \\ last_{i,j} = SW_{first_i}}} C_j + \sum_{\substack{m \\ first_m \neq first_i \\ first_{i,m} \neq SW_{first_i}}} C_m - T_{min} \right) + \sum_{\substack{m \\ first_m \neq first_i \\ first_{i,m} \neq SW_{first_i}}} \left\lfloor \frac{A_{i,m}}{T_m} \right\rfloor \cdot C_m$$

Moreover, TA considers that flows postponing the studied one are all processed on the slowest node (denoted *slow*). The utilization factor U_{slow} on this node is given in Eq. (4.10) and is less than 1.

$$U_{slow} = \sum_{\substack{i \cup j \\ P_i \cap P_j \neq \emptyset}} \frac{C_j}{T_j} \geq \sum_{\substack{i \cup j \\ P_i \cap P_j \neq \emptyset}} \frac{C_j}{T_{max}} \tag{4.10}$$

If the sum of the processing times of flows delaying the studied flow on its first crossed switch is greater or equal to the total processing time of those activated on the same source and sharing the same output link with τ_i on SW_{first_i} (see Eq. (4.11)), then the factor multiplied by K present in the expression of $g(KT_{min})$ is smaller or equal to $U_{slow} \cdot T_{max} - T_{min}$.

$$\sum_{\substack{k \\ first_{i,k} = SW_{first_i}}} C_k \geq \sum_{\substack{j \\ first_j = first_i \\ last_{i,j} \neq SW_{first_i}}} C_j \tag{4.11}$$

Using Eq. (4.10) and Eq. (4.11), $g(KT_{min})$ can be bounded as follows.

$$g(KT_{min}) \leq K\left(U_{slow} \cdot T_{max} - T_{min}\right) + \sum_{\substack{m \\ first_m \neq first_i \\ first_{i,m} \neq SW_{first_i}}} \left\lfloor \frac{A_{i,m}}{T_{min}} \right\rfloor \cdot C_m$$

$$\leq K \cdot T_{max}\left(U_{slow} - \frac{T_{min}}{T_{max}}\right) + \sum_{\substack{m \\ first_m \neq first_i \\ first_{i,m} \neq SW_{first_i}}} \left\lfloor \frac{A_{i,m}}{T_{min}} \right\rfloor \cdot C_m$$

$g((KT_{min})$ is negative if Eq. (4.12) and Eq. (4.13)are verified.

$$U_{slow} - \frac{T_{min}}{T_{max}} < 0 \tag{4.12}$$

$$A_{i,m} < T_{min} \tag{4.13}$$

– The expression of $g(KT_{min} - A_{i,m})$ can be written as follows.

$$g(KT_{min} - A_{i,m}) = 2 \sum_{\substack{i \bigcup j \\ first_j = first_i \\ last_{i,j} \neq SW_{first_i}}} \left\lfloor \frac{KT_{min} - A_{i,m}}{T_{min}} \right\rfloor C_j + \sum_{\substack{j \\ first_j = first_i \\ last_{i,j} = SW_{first_i}}} \left\lfloor \frac{KT_{min} - A_{i,m}}{T_{min}} \right\rfloor C_j + \sum_{\substack{m \\ first_m \neq first_i \\ first_{i,m} \neq SW_{first_i}}} \left\lfloor \frac{KT_{min}}{T_{min}} \right\rfloor C_m - (KT_{min} - A_{i,m})$$

$$
\begin{aligned}
&= 2 \sum_{\substack{i \bigcup j \\ first_j = first_i \\ last_{i,j} \neq SW_{first_i}}} \left\lfloor K - \frac{A_{i,m}}{T_{min}} \right\rfloor C_j + \sum_{\substack{j \\ first_j = first_i \\ last_{i,j} = SW_{first_i}}} \left\lfloor K - \frac{A_{i,m}}{T_{min}} \right\rfloor C_j \\
&+ \sum_{\substack{m \\ first_m \neq first_i \\ first_{i,m} \neq SW_{first_i}}} \left\lfloor \frac{KT_{min}}{T_{min}} \right\rfloor C_m - (KT_{min} - A_{i,m}) \\
&= K \left(2 \sum_{\substack{i \bigcup j \\ first_j = first_i \\ last_{i,j} \neq SW_{first_i}}} C_j + \sum_{\substack{j \\ first_j = first_i \\ last_{i,j} = SW_{first_i}}} C_j + \right. \\
&\left. \sum_{\substack{m \\ first_m \neq first_i \\ first_{i,m} \neq SW_{first_i}}} C_m - T_{min} \right) \\
&+ 2 \sum_{\substack{i \bigcup j \\ first_j = first_i \\ last_{i,j} \neq SW_{first_i}}} \left\lfloor \frac{-A_{i,m}}{T_{min}} \right\rfloor C_j + \sum_{\substack{j \\ first_j = first_i \\ last_{i,j} = SW_{first_i}}} \left\lfloor \frac{-A_{i,m}}{T_{min}} \right\rfloor C_j + A_{i,m}
\end{aligned}
$$

Similar to $g(KT_{min})$, the expression of $g(KT_{min} - A_{i,m})$ can be bounded as follows.

$$
\begin{aligned}
g(KT_{min} - A_{i,m}) \leq K \left(U_{slow} \cdot T_{max} - T_{min} \right) + A_{i,m} \\
+ 2 \sum_{\substack{i \bigcup j \\ first_j = first_i \\ last_{i,j} \neq SW_{first_i}}} \left\lfloor \frac{-A_{i,m}}{T_{min}} \right\rfloor C_j + \sum_{\substack{j \\ first_j = first_i \\ last_{i,j} = SW_{first_i}}} \left\lfloor \frac{-A_{i,m}}{T_{min}} \right\rfloor C_j
\end{aligned}
$$

Furthermore, we have shown that $A_{i,m}$ should be less than T_{min} and $U_{slow} \cdot T_{max} - T_{min}$ should be negative. Hence, we have the following:

$$g(KT_{min} - A_{i,m}) \leq K \cdot (U_{slow} \cdot T_{max} - T_{min}) + A_{i,m}$$
$$\leq (U_{slow} \cdot T_{max} - T_{min}) + A_{i,m}$$

It is clear that $g(KT_{min} - A_{i,m})$ is negative if $A_{i,m}$ is less than $T_{min} - U_{slow} \cdot T_{max}$.

– The value of $g(0)$

$$g(0) = \sum_{\substack{m \\ first_m \neq first_i \\ first_{i,m} \neq SW_{first_i}}} \left\lfloor \frac{A_{i,m}}{T_{min}} \right\rfloor \cdot C_m$$

We know that $A_{i,m}$ is less than $T_{min} - U_{slow} \cdot T_{max}$. This means that the value of $g(0)$ is null.

To conclude, the function $f(t)$ is less or equal to zero and $f(0) = 0$. Hence, under these conditions, the maximum of $f(t)$ is reached for $t = 0$. □

Property 4.1. *Let τ_m be the set of flows delaying the studied flow denoted τ_i after its first crossed switch.*
If 1) the sum of the processing times of flows delaying the studied flow on its first crossed switch is greater or equal to the total processing time of those activated on the same source and sharing the same output link with τ_i on SW_{first_i} and, 2) $A_{i,m}$ is less than $T_{min} - U_{slow} \cdot T_{max}$, then an upper bound of the end-to-end response time is equal to:

$$R_i \leq (|P_i| - 1)\, L + \sum_{\substack{h \in P_i \\ h \neq last_i}} \left(\max_{j; h \in P_j} \{C_j\} \right) + \max_{h = SW_{first_i}} \{C_k\} - \min_{h = SW_{first_i}} \{C_j\}$$
$$+ 2 \sum_{\substack{i \bigcup j \\ first_j = first_i \\ last_{i,j} \neq SW_{first_i}}} C_j + \sum_{\substack{j \\ first_j = first_i \\ last_{i,j} = SW_{first_i}}} C_j + \sum_{\substack{m \\ first_m \neq first_i \\ first_{i,m} \neq SW_{first_i}}} C_m$$

4.2.3 All flows have the same processing time and the same period

In this section, we consider that flows have all the same processing time and the same period. Let C be the processing time and T the period.
We first prove that, under these hypothesis, the value of $A_{i,m}$ is smaller than U_{slow} times the period. Then, we show that, for flows respecting a specific utilization factor's condition, an upper bound of the end-to-end response time can be easily derived.

4.2.3.1 Expression of $A_{i,m}$

The expression of $A_{i,m}$ differs whether τ_m (the flow postponing the studied flow after the first crossed switch) delay τ_i on the last switch or on another one.
For simplicity sake, we note $first\_SW$ the first switch, $SW+l$ the switch positioned at l hops from SW and τ_{m_l} are flows postponing τ_i on $SW+l$ (refer to Fig. 4.3). Let $last\_SW$ be the last switch crossed by τ_i.

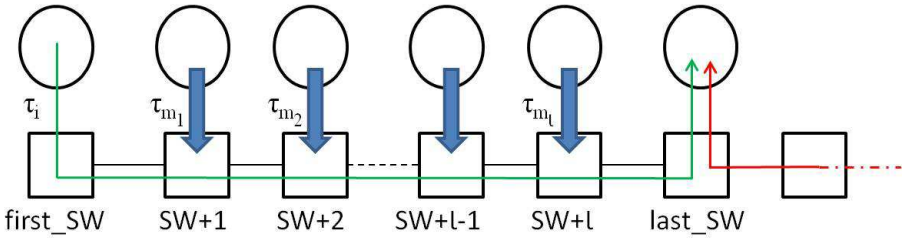


Figure 4.3: Representative figure

We start by determining the expression of $A_{i,m}$ on node $SW+1, SW+2, ..., SW+l$ then on the last switch.
On node $SW+1$, A_{i,m_1} is equal to the sum of jitters of flow τ_i and τ_{m_1}. We recall

that the jitter of a flow is equal to the maximum arrival time S_{max} minus the minimum arrival time S_{min}. Hence, S_{max} of τ_{m_1} is simply equal to its response time on the source $+L$. The response time is equal to the number of flows activated on the source of τ_{m1} times the processing time C. Furthermore, S_{min} can be computed by considering that there are no other flows in the network. Thus, S_{min} of τ_{m_1} is equal to $C + L$. Similarly, S_{min} of τ_i is equal to $2C + 2L$. To compute S_{max_i} on $SW + 1$, we need to determine its response time on the previous switch (i.e $first\_SW$). It is given in the equation below and A_{i,m_1} is given in Eq. (4.16).

$$
\begin{aligned}
S_{max_i}^{SW+1} &= R_i^{first\_SW} + L \\
&= 2 \cdot \sum_{\substack{i \bigcup j \\ first_j = first_i \\ last_{i,j} \neq SW_{first_i}}} C + \sum_{\substack{j \\ first_j = first_i \\ last_{i,j} = SW_{first_i}}} C + (C + L) + L \\
&\quad + \max_{t \geq 0} \left\{ 2 \cdot \sum_{\substack{i \bigcup j \\ first_j = first_i \\ last_{i,j} \neq SW_{first_i}}} \left\lfloor \frac{t}{T} \right\rfloor \cdot C + \sum_{\substack{j \\ first_j = first_i \\ last_{i,j} = SW_{first_i}}} \left\lfloor \frac{t}{T} \right\rfloor \cdot C - t \right\}
\end{aligned}
$$

The maximum of the previous formula is obtained for $t = 0$.

Proof. Let us consider the following function f(t).

$$
f(t) = 2 \cdot \sum_{\substack{i \bigcup j \\ first_j = first_i \\ last_{i,j} \neq SW_{first_i}}} \left\lfloor \frac{t}{T} \right\rfloor \cdot C + \sum_{\substack{j \\ first_j = first_i \\ last_{i,j} = SW_{first_i}}} \left\lfloor \frac{t}{T} \right\rfloor \cdot C - t
$$

To obtain the maximum, we need to test the instants $t = KT$.

$$f(KT) = K \left(2 \cdot \sum_{\substack{i \bigcup j \\ first_j = first_i \\ last_{i,j} \neq SW_{first_i}}} C + \sum_{\substack{j \\ first_j = first_i \\ last_{i,j} = SW_{first_i}}} C - T \right)$$

On the other hand, the Trajectory Approach considers that all flows postponing the studied flow are processed on the slowest node of the path of the studied flow. Hence, To be able to apply the approach, the utilization factor on this node denoted U_{slow} should be less than one and its expression is given (4.14).

$$U_{slow} = \sum_{\substack{i \bigcup j \\ first_j = first_i}} \frac{C}{T} + \sum_{k; first_{i,k} = SW_{first_i}} \frac{C}{T} \sum_{\substack{m; first_m \neq first_i \\ first_{i,m} \neq SW_{first_i}}} \frac{C}{T} \tag{4.14}$$

If the number of flows postponing the studied flow on SW_{first_i} is greater than the number of flows competing with the studied flow over resources on its source and SW_{first_i} (see (4.15)), then $f(KT)$ is smaller or equal to $K \cdot T(U_{slow} - 1)$. This means that the maximum of the previously defined function $f(t)$ is reached for $K = 0$.

$$\sum_{\substack{k \\ first_{i,k} = SW_{first_i}}} C \geq \sum_{\substack{i \bigcup j \\ first_j = first_i \\ last_{i,j} \neq SW_{first_i}}} C \tag{4.15}$$

□

The expressions of S_{max_i} and A_{i,m_1} become:

$$S_{max_i}^{SW+1} = 2 \cdot \sum_{\substack{i \bigcup j \\ first_j = first_i \\ last_{i,j} \neq SW_{first_i}}} C + \sum_{\substack{j \\ first_j = first_i \\ last_{i,j} = SW_{first_i}}} C + (C + L) + L$$

$$A_{i,m_1} = \left(S^{SW+1}_{max_i} - S^{SW+1}_{min_i}\right) + \left(S^{SW+1}_{max_{m_1}} - S^{SW+1}_{min_{m_1}}\right)$$

$$= \left(2 \cdot \sum_{\substack{i \bigcup j \\ first_j = first_i \\ last_{i,j} \neq SW_{first_i}}} C + \sum_{\substack{j \\ first_j = first_i \\ last_{i,j} = SW_{first_i}}} C - C \right) + \left(\sum_{m_1} C - C \right)$$

$$A_{i,m_1} = 2 \sum_{\substack{i \bigcup j \\ first_j = first_i \\ last_{i,j} \neq SW_{first_i}}} C + \sum_{\substack{j \\ first_j = first_i \\ last_{i,j} = SW_{first_i}}} C + \sum_{m_1} C - 2C \tag{4.16}$$

Since A_{i,m_1} is less than $U_{slow} \cdot T$ then it is also less than T on $SW + 1$. Similarly, the expression of A_{i,m_2} on node $SW + 2$ is given in Eq.(4.17) and it is smaller than $U_{slow} \cdot T$.

$$A_{i,m_2} = 2 \sum_{\substack{i \bigcup j \\ first_j = first_i \\ last_{i,j} \neq SW_{first_i}}} C + \sum_{\substack{j \\ first_j = first_i \\ last_{i,j} = SW_{first_i}}} C + \sum_{m_1} C + \sum_{m_2} C - 2C \tag{4.17}$$

Moreover, the expression of A_{i,m_l} on node $SW + l$ is given in Eq. (4.18). Based on Eq.(4.15), it is clear that A_{i,m_l} is less than $U_{slow} \cdot T$.

$$A_{i,m_l} = 2 \sum_{\substack{i \bigcup j \\ first_j = first_i \\ last_{i,j} \neq SW_{first_i}}} C + \sum_{\substack{j \\ first_j = first_i \\ last_{i,j} = SW_{first_i}}} C + \sum_{m_1} C + \sum_{m_2} C + ... + \sum_{m_l} C - 2C \tag{4.18}$$

It is clear that for every flow τ_{m_x} $(x = [1, 2, ..., l])$ meeting τ_i for the first time on a switch between the first and the last switch crossed by τ_i, the sum of jitters A_{i,m_x} is always less than $U_{slow} \cdot T$.

We need to determine now the expression of $S_{max} - S_{min}$ of both the studied flow and the flow τ_m postponing it on the last switch. The expression of $S_{max} - S_{min}$ of τ_i is simple to obtain; it is equal to:

$$S_{max_i}^{last\_SW} - S_{min_i}^{last\_SW} = 2 \sum_{\substack{i \bigcup j \\ first_j = first_i \\ last_{i,j} \neq SW_{first_i}}} C + \sum_{\substack{j \\ first_j = first_i \\ last_{i,j} = SW_{first_i}}} C + \sum_{m_1} C + \sum_{m_2} C + ... + \sum_{m_l} C - C$$

The reasoning on node $last\_SW$ is slightly different when it comes to computing the maximum arrival time S_{max} of the flow τ_m. In fact, before reaching the last switch, this flow might have crossed several switches and might have been delayed by several other flows. However, the value of S_{max} is still equal to $R_m^{pre_m(last\_SW)}$ its response on the previous node $+L$. The same logic used to compute R_i on node $SW + l$ can be used to determine $R_m^{pre_m(last\_SW)}$ which is equal to the sum of the delay introduced by:

- flows activated on the source of τ_m multiplied by 2 such that the last common node between these flows and τ_m is not the source of τ_m.
- flows activated on the source of τ_m such that the last common node between these flows and τ_m is the source of τ_m.
- flows postponing τ_m on a switch different than the first switch crossed by τ_m.

By analogy, the expression of $S_{max_m} - S_{min_m}$ on $last\_SW$ is similar to that of $S_{max_i} - S_{min_i}$ on this node.

TA considers that all flows postponing τ_i are processed on the slowest node. Let us assume that it is $last\_SW$. On the other hand, flows postponing τ_m are also supposed to be processed on the slowest node; let it be also $last\_SW$. And since the utilization factor on the latter node should be less than one, therefore $A_{i,m}$ which includes the delay introduced by flows τ_i, τ_m and those postponing them should also be less than or equal to the corresponding utilization factor on the slowest node multiplied by the period.

Property 4.2. *When all flows have the same processing time C and the same period T, the value of $A_{i,m}$ of flows postponing the studied flow after the first crossed switch is smaller than $U_{slow} \cdot T$.*

4.2.3.2 End-to-end response time

In section 4.2.2, an upper bound of the end-to-end response time was derived for flows verifying that the terms $A_{i,m}$ is smaller than $T(1 - U_{slow})$. We recall that index m refers to flows meeting the studied for the first time on a node different than the source and the first switch. The corresponding upper bound can be written as follows.

$$R_i \leq (|P_i| - 1)(L + C) + 2 \sum_{\substack{i \bigcup j \\ first_j = first_i \\ last_{i,j} \neq SW_{first_i}}} C + \sum_{\substack{j \\ first_j = first_i \\ last_{i,j} = SW_{first_i}}} C + \sum_{\substack{m \\ first_m \neq first_i \\ first_{i,m} \neq SW_{first_i}}} C$$

On the other hand, we have shown that when all flows have the same processing time and the same period, the terms $A_{i,m}$ becomes less than $U_{slow} \cdot T$.
Hence, to be able to use the result obtained in section 4.2.2, the condition presenented

in Eq. (4.19) should be respected. In other words, the utilization factor on the slowest node should be less than 1/2.

$$A_{i,m} \leq U_{slow} \cdot T \leq T(1 - U_{slow}) \tag{4.19}$$

Property 4.3. *When all flows have the same processing time C and the same period T, if the utilization factor on the slowest node is smaller than $1/2$, then an upper bound of the end-to-end response time is equal to :*

$$R_i \leq (|P_i| - 1)(L + C) + 2 \sum_{\substack{i \bigcup j \\ first_j = first_i \\ last_{i,j} \neq SW_{first_i}}} C + \sum_{\substack{j \\ first_j = first_i \\ last_{i,j} = SW_{first_i}}} C + \sum_{\substack{m \\ first_m \neq first_i \\ first_{i,m} \neq SW_{first_i}}} C$$

4.3 Results on industrial configurations

In this section, we compare the Scalable Trajectory Approach (STA) and the Enhanced Trajectory Approach (ETA) in terms of pessimism of the upper bounds and total computation runtime. We have also studied the effect of the variation of the processing time on the upper bound and the effect of the variation of the period on the runtime.

For this purpose, we have developed three programs under C++ that computes the upper bounds using:

- the original version of the Trajectory Approach
- the Enhanced version of the Trajectory Approach
- the Scalable Trajectory Approach.

Each program takes a text file (.txt) in its input. Each line of the file contains information about the exchanged flows such as their processing time, their period and the set of nodes crossed by each flow. The output file contains the end-to-end response time's upper bound of each flow and the time required to compute it.
The program using STA proceeds as follows: for flows verifying the conditions listed above, STA is directly applied otherwise ETA is used to compute the upper bound.

4.3.1 Comparison between ETA and STA results

We first apply only ETA then STA on a network composed of 10 switches. We are interested in comparing the obtained results using these two approaches in terms of precision of the upper bounds and the total runtime. One thousand flows are exchanged across the chosen network. The sources and destinations were chosen randomly. All flows have the same processing time which is equal to 26μs and the same period ($T = 100$ms). In addition, the switching delay is known and is equal to 3μs. We use the developed programs to analyze this network; the upper bounds are presented in Fig. 4.4. We state that the obtained results are ordered in an increasing order.
When using the STA program, we notice that on 71.9% of the exchanged flows satisfies the previously described condition and thus the upper bounds of these flows are obtained instantaneously. For the rest of the flows (i.e. 28.2%), the condition is not satisfied and their upper bounds are obtained using ETA. Moreover, for flows satisfying the condition, the average error between the bound obtained using STA and that using ETA is 1.75%. In addition, when using the STA program, the results show that, for 52% of the exchanged flows, the bounds obtained using STA are equal to the bounds obtained using ETA.

Nevertheless, in order to compute the upper bounds of the one thousand flows, ETA program puts **four days** whiles STA program requires only 15 **seconds**.

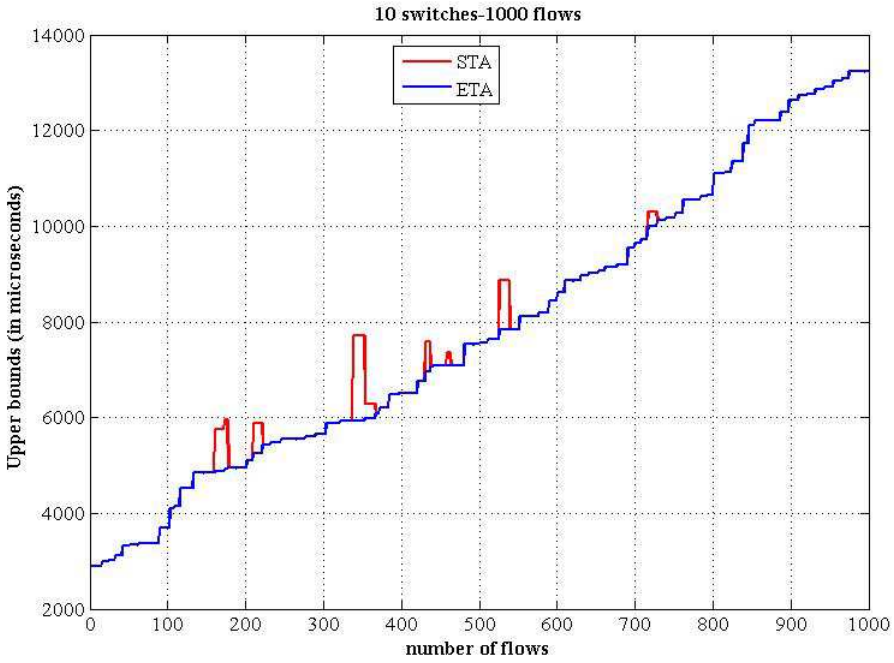
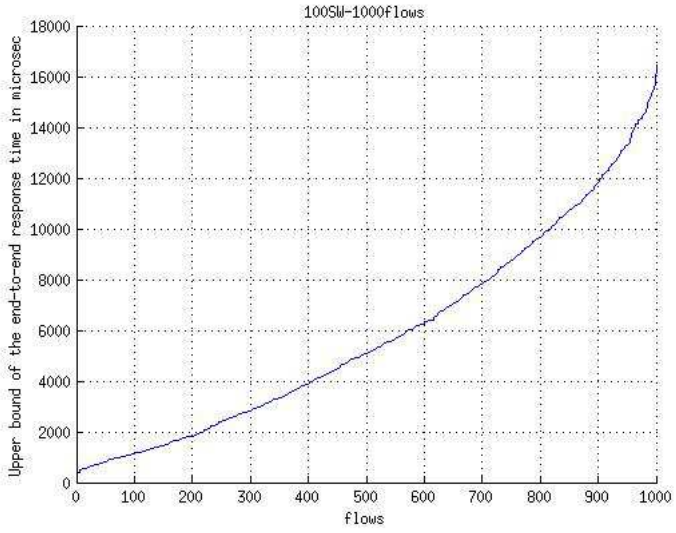
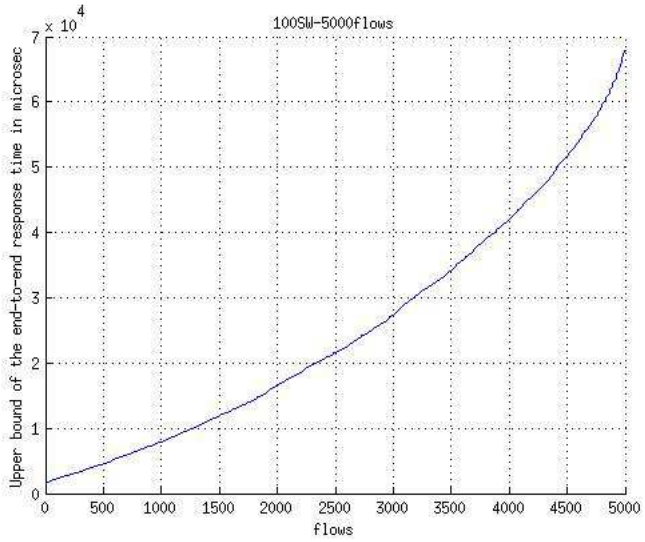


Figure 4.4: Comparison between ETA and STA results

For more realistic analysis, we have also applied the proposed approach on a large network composed of one hundred switches. Flows have the same characteristics as listed before. The upper bounds when the number of flows exchanged is equal 1000 and 5000 flows are respectively illustrated in Fig. 4.5(a) and Fig 4.5(b). STA requires about 45 minutes to determine the upper bounds for the one thousand flows. However, it takes 11 hours to compute the upper bounds of the 5000 flows.



(a) number of flows=1000



(b) number of flows=5000

Figure 4.5: Upper bounds determined by STA for a network composed of (a) 1000 and (b) 5000 flows

We have also studied the three methods (the Trajectory, the Enhanced Trajectory and the scalable Trajectory Approaches) in terms of total runtime as shown in Fig 4.6. In this analysis, the network is composed of ten switches and the number of flows varies from 100 to 1000. In each scenario, the sources and the destinations are chosen randomly. The processing time is equal to $1\mu s$ and the flow's period is equal to $1ms$. In addition, the switching delay is considered to be null.

As the number of flows increases, the three methods requires additional time to upper bound all flows. However, the total runtime required by ETA grows exponentially, while that required by TA grows linearly. Fig 4.6 shows a zoomed view of the results obtained using STA. In this case, the total runtime increases slowly. For even a configuration composed of one thousand flows, STA puts only 5 seconds to compute the upper bounds of all flows.

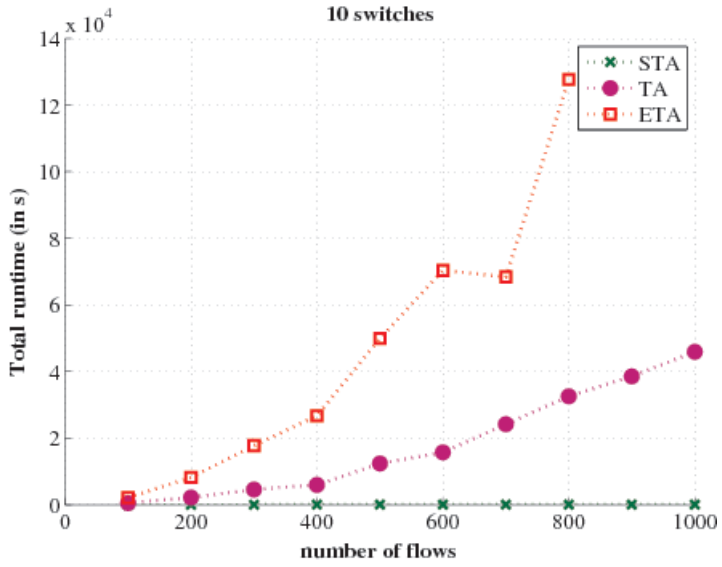


Figure 4.6: Total runtime using STA, ETA and TA

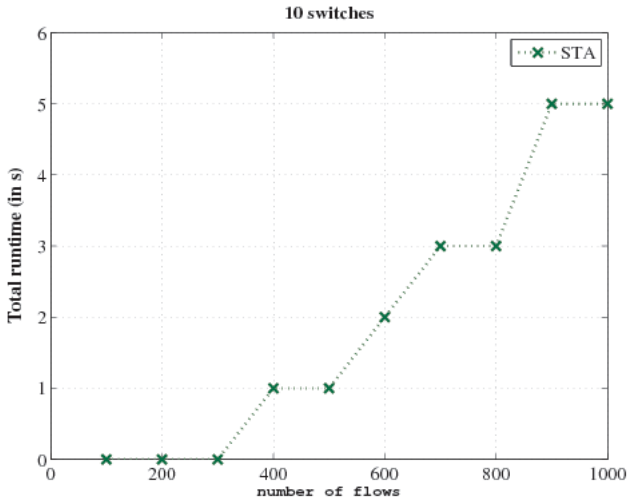


Figure 4.7: Total runtime required to analyze a network composed of 10 switches using STA

4.3.2 Effect of the variation of the processing time on the upper bound

One thousand flows are exchanged across the studied network which is composed of one hundred identical switches. Their sources and the destinations are chosen randomly.

For this simulation, we consider that all flows have the same period and is fixed to 100 msec. In addition, all flows have the same processing time. Then, the processing time is varied from $C = 30\mu$sec to 60μsec. We are interested in studying the effect of the processing time changes on the computed upper bound. The results presented in Fig. 4.8 shows that, for the same flow, the upper bound increases with the processing time.

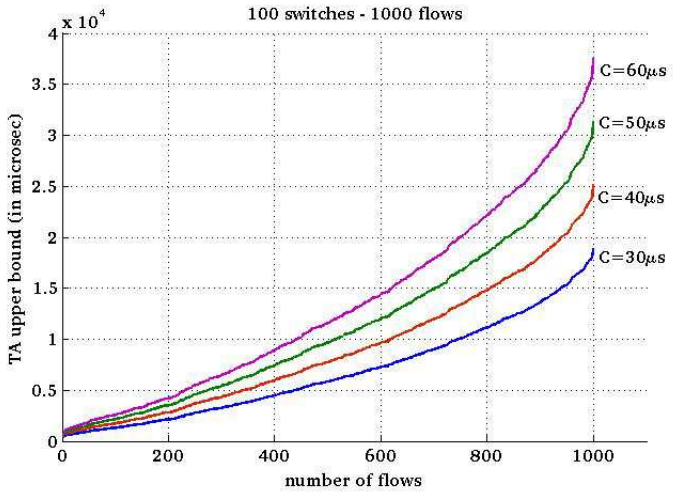


Figure 4.8: Variation of the upper bound with the processing time.

4.3.3 Effect of the variation of the period on the runtime

The same configuration described in the previous section is used to study the effect of the period on the total runtime. In this analysis, the processing time of the flows is fixed to 26μseconds. We increase the period of the flows starting from 30 milliseconds until reaching 100milliseconds.

Results given in Fig. 4.9 show the impact of the flow's period on the total runtime. For a period varying from $T = 30ms$ to $80ms$, the time required to compute the upper bounds of the one thousand flows is approximately the same and is around 2 seconds. When we increase the flow's period to $90ms$, the runtime becomes bigger and is equal to 30 minutes. For a period up to $100ms$, the total computational runtime is equal to 45 minutes.

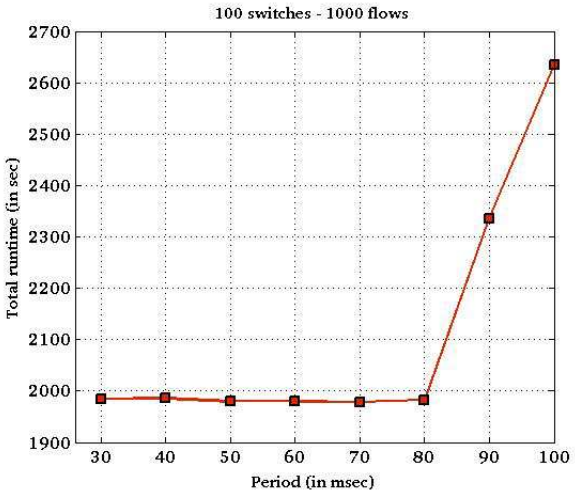


Figure 4.9: Variation of the total runtime in function of the period.

4.3.4 Flows have different processing times and periods

In this section, we are interesting in analyzing the case where flows exchanged across the studied network do not have neither the same processing times nor the same periods.

The network understudy is composed of one hundred switches connected in a line topology. The number of exchanged flows is up to one thousand. The sources and the destinations are chosen randomly. The processing times varies between 25 and 50μsec, while the periods are equal to either 0.05 or 0.1msec.

For each flow, we test if the conditions to apply the scalable approach are satisfied. If not, the enhanced approach is used to determine the upper bound. The results illustrated in Fig. 4.10 are presented in an increasing order.

Results regarding runtime and are grouped in Table 4.2. For 97.2% of the flows, the Scalable Trajectory Approach was used to determine an upper bound; while the

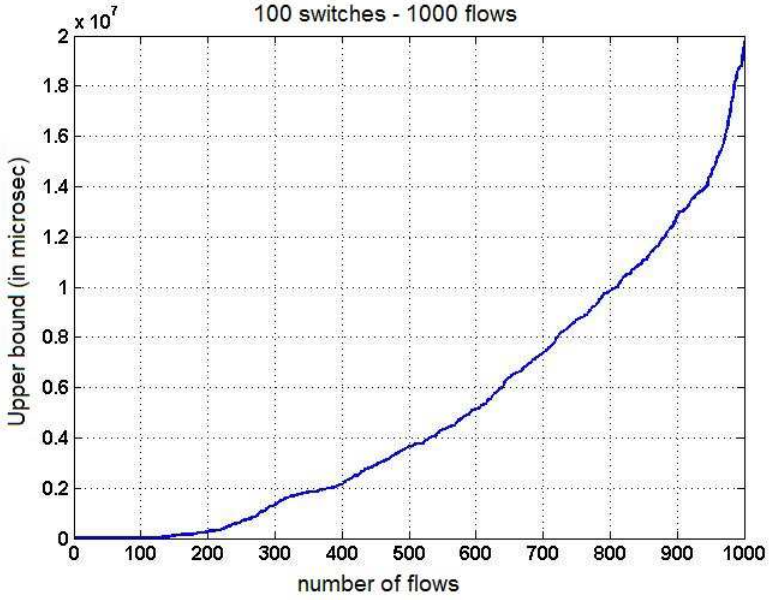


Figure 4.10: Upper bound in microseconds

enhanced Trajectory Approach was applied for the rest. Moreover, the total runtime taken by STA to calculate the upper bounds of 972 flows was approximately up to 2.4 hours. On the other hand, ETA needed up to 26 hours to determine the upper bounds of just 28 flows.

	STA	ETA	total
number of flows	972	28	1000
runtime (in hours)	2.4	26	28.4

Table 4.2: Comparison results between STA and ETA

4.4 Conclusion

In this chapter, we have tackled the scalability problem confronted when applying the Trajectory Approach on a large industrial configuration. We have proposed a method allowing to compute immediately an upper bound for some flows. These flows have to satisfy specific conditions. We have also investigated the case for which all flows have the same processing time and the same period. In this particular case, we proved that the condition becomes simply a load condition.

We have applied this approach on a limited industrial network. Results show that computing the upper bounds is faster using our solution without significant loss in the precision of the obtained bound. Hence, our solution allows us obtaining results on large industrial configuration where the number of variables to determine is huge.

Chapter 5

Conclusions and perspectives

5.1 Conclusions

System verification and validation is an old addressed topic. The studied network is an Ethernet-switched based network. Though the use of switched-Ethernet eliminates the indeterminism introduced by the CSMA/CD protocol, yet the problem is shifted to the buffer level. In fact, the flows compete over the resources inside the switch and the essential problem consists of determining the time spent by a packet in the buffer. Hence, this thesis focuses on timing analysis.

First, the methods that can be used to perform timing analysis were briefly explored in chapter 2. In particular, we are asked to compute an upper bound of the end-to-end response time of any flow exchanged across the studied network. These methods can be divided into two groups: the simulation-based and the analytical approaches. We recall that, when interested in computing deterministic guarantees, simulation approaches fail to meet the requirements. Hence, approaches based on

mathematical models sound like good candidates. Among the analytical approaches, we have the Model Checking (MC), the Network Calculus (NC) and the Trajectory Approach(TA). Model checking based on timed automata computes the exact worst-case response time and offers the corresponding scenario. However, it is still unable to analyze large networks. While the Network Calculus is being used in the system verifcation and validation process nowadays, the corresponding upper bound of the end-to-end response time is pessimistic which leads to overdimensioning the network. Another possible method is the Trajectory Approach; it uses results established by the scheduling theory to derive an upper bound. Both the network calculus and the Trajectory Approach were applied on an avionic system; the results showed that TA outperforms NC. For all these reasons, we focused our study on the Trajectory Approach.

Secondly, the limitations of applying the Trajectory Approach for FIFO scheduled flows were discussed in chapter 3. The first one is the precision of the upper bound. We have identified the flow's configurations that introduce pessimism into the computed upper bound. The second one is the scalability of the approach. We have shown that the Enhanced Trajectory Approach requires more time than the Trajectory Approach to analyze the same configuration. As much as the precision of the upper bound is important, the scalability of the method is even more important. For that, we are interested in computing an upper bound of the end-to-end response time but in a reduced time frame.

Finally, the scalability problem confronted when applying the Trajectory Approach on a large industrial configuration was tackled in chapter 4. We have proposed a method allowing to compute immediately an upper bound for some flows. These

flows have to satisfy specific conditions. We have also investigated the case for which all flows have the same processing time and the same period. In this particular case, we proved that the condition becomes simply a load condition. We have applied this approach on a limited industrial network. Results show that computing the upper bounds is faster using our solution without significant loss in the precision of the obtained bound. Hence, our solution allows us to obtain results on large industrial configuration where the number of parameters to determine is huge.

To summarize, our work consists of using the Trajectory Approach to derive the upper bounds of the end-to-end response time of every flow present in a large network.

5.2 Perspectives

There are several prospects of the work performed in this thesis:

- The scalable Trajectory Approach was applied on a line topology. It is also interesting on other topologies such as a tree network for example.
- It is also interesting to compare the results obtained using the Scalable Trajectory Approach and those obtained using the Network Calculus.
- Another axis consists of applying the Scalable Trajectory Approach on networks holding heterogeneous traffic. In this case for example, flows can be attributed a fixed priority and those having the same priority can be scheduled according to their arrival time.
- Another topic to research is to find a compromise between the computation

complexity and the precision of the upper bound. For flows that do not satisfy the condition of the Scalable Trajectory Approach, it would be interesting to compute an upper bound using another approach even if this bound is pessimistic. Then after comparing this bound to the deadline, if the deadline is not respected, then the computed bound could be tighten. So on until the deadline is respected.

Bibliography

[1] H. Petroski, *To Engineer is Human: The Role of Failure in Successful Design*. New York: Vintage, 1992.

[2] A. T. Bahill and S. J. Henderson, "Requirements development, verification and validation exhibited in famous failures," *System Engineering*, pp. 1–14, 2005.

[3] M. Talbott, "Why system fails (viewed from hindsight)," in *In Proceeding of the International Conference on System Engineeering INCOSE*.

[4] D. Dorner, *The Logic Of Failure: Recognizing And Avoiding Error In Complex Situations*. Basic Books, 1 ed., Aug. 1997.

[5] Y. Bar-Yam, "When systems engineering fails-toward complex systems engineering," in *International Conference on In Systems, Man and Cybernetics*.

[6] D. Trognon, "Control network case study for dening a safety qualification method," tech. rep., EDF, 2007.

[7] H. Bauer, *Analyse pire cas de flux hétérognes dans un réseau embarqué avion*. PhD thesis, Institut de Recherche en Informatique de Toulouse, 2011.

[8] S. Medlej, S. Martin, and J.-M. Cottin, "Identifying sources of pessimism in the trajectory approach with FIFO scheduling," in *Embedded Real-Time Software and Systems ERTS2*, 2012.

[9] M. T. J. Jasperneite, P. Neumann and K. Watson, "Deterministic real time communication with switched ethernet," in *4th international workshop on Factory communication systems.*

[10] "Pspice." http://www.cadence.com/products/orcad/pspice-simulation/Pages/default.aspx.

[11] "Webnsm simulation tool kit." http://www.webnms.com/simulator/network-simulator-ds.html.

[12] "Matlab." http://www.mathworks.fr/.

[13] S. Perathoner, E. Wandeler, L. Thieler, A. Hamann, S. Schliecker, R. Henia, R. Racu, R. Ernst, and M. G. Harbour, "Influence of different abstraction on the performance analysis of distributed hard-real time systems," in *Proceedings of the 7th ACM and IEEE international conference on Embedded Softaware*, (NY, USA), 2007.

[14] E. Clarke, O. Grumberg, and D. A. Peled, *Model Checking.* MIT Press, Cambridge, MA, USA, 2000.

[15] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical Computer Science*, vol. 126, pp. 183–235, 1994.

[16] K. G. Larsen, P. Pettersson, and W. Yi, "UPPAAL in a Nutshell," *Int. Journal on Software Tools for Technology Transfer*, vol. 1, pp. 134–152, October 1997.

[17] B. Berard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, and P. Schnoebelen, *Systems and Software Verification: Model-Checking Techniques and Tools*. Springer Publishing Company, Incorporated, 1st ed., 2010.

[18] C. Baier, B. Haverkort, H. Hermanns, and J. pieter Katoen, "Automated performance and dependability evaluation using model checking," in *In Performance Evaluation of Complex Systems: Techniques and Tools, Performance 2002, Tutorial Lectures*, pp. 261–289, SpringerVerlag, 2002.

[19] "Uppaal." http://www.uppaal.com/.

[20] "Blast." http://mtc.epfl.ch/software-tools/blast/index-epfl.php.

[21] "Spin." http://spinroot.com/spin/whatispin.html.

[22] Z. Xin-feng, W. Jian-dong, L. Bin, Z. Jun-wu, and W. Jun, "Methods to tackle state explosion problem in model checking," in *Proceedings of the 2009 Third International Symposium on Intelligent Information Technology Application - Volume 02*, IITA '09, (Washington, DC, USA), pp. 329–331, IEEE Computer Society, 2009.

[23] E. M. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith, "Progress on the state explosion problem in model checking," in *Informatics - 10 Years Back. 10 Years Ahead.*, (London, UK), pp. 176–194, Springer-Verlag, 2001.

[24] M. Li and P. M. Vitnyi, *An Introduction to Kolmogorov Complexity and Its Applications*. Springer Publishing Company, Incorporated, 3 ed., 2008.

[25] R. L. Cruz, "a calculus for network delay, part i: network elements in isolation," *IEEE transactions on Information theory*, 1991.

[26] R. L. Cruz, "a calculus for network delay, part ii: network analysis," *IEEE transactions on Information theory*, 1991.

[27] R. L. Cruz, "Quality of service guarantees in virtual circuit switched networks," *IEEE Journal on selected areas in communications*, vol. 13, pp. 1048–1056, 1995.

[28] R. L. Cruz, "Sced+: Efficient management of quality of service guarantees," in *In Proceedings of INFOCOM'98*, pp. 625–642, 1998.

[29] R. Agrawal and R. Rajan, "Performance bounds for guaranteed and adaptive services," tech. rep., IBM, 1996.

[30] C.-S. Chang, *Performance Guarantees in Communication Networks*. London, UK: Springer-Verlag, 2000.

[31] C.-S. Chang, "Stability, queue length and delay of deterministic and stochastic queueing networks," *IEEE Transactions on Automatic Control*, vol. 39, pp. 913–931, 1994.

[32] J.-Y. L. Boudec, "Application of network calculus to guaranteed service networks," *IEEE Transaction on Information Theory*, vol. 44, pp. 1087–1096, Sept. 2006.

[33] J.-Y. L. Boudec and P. Thiran, *Network calculus: a theory of deterministic queuing systems for the internet*. Berlin, Heidelberg: Springer-Verlag, 2001.

[34] M. Fidler, "Survey of deterministic and stochastic service curve models in the network calculus," *IEEE Communication Surveys and Tutorials*, vol. 12, pp. 59–86, Jan. 2010.

[35] H. Schioler, H. P. Schwefel, and M. B. Hansen, "Cync: A matlab/simulink toolbox for network calculus," October 2007.

[36] L. Thiele, S. Chakraborty, and M. Naedele, "Real-time calculus for scheduling hard real-time systems," in *Proc. international symposium on Circuits and systems*, (Geneva), 28-31 May 2000.

[37] N. Gollan, F. A. Zdarsky, I. Martinovic, and J. B. Schmitt, "The disco network calculator," in *in Proceeding Measuring, Modelling and Evaluation of Computer and Communication Systems (MMB), 14th GI/ITG Conference*.

[38] J. B. Schmitt and F. A. Zdarsky, "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), ACM, 2006.

[39] A. Bouillard and E. Thierry, "An algorithmic toolbox for network calculus," *Discrete Event Dynamic Systems*, pp. 3–49, 2008.

[40] L. Bisti, L. Lenzini, E. Mingozzi, and G. Stea, "Deborah: a tool for worst-case analysis of fifo tandems," in *Proceedings of the 4th international conference on Leveraging applications of formal methods, verification, and validation - Volume Part I*, ISoLA'10, (Berlin, Heidelberg), pp. 152–168, Springer-Verlag, 2010.

[41] R. Henia, A. Hamann, M. Jersak, R. Racu, K. Richter, and R. Ernst, "System level performance analysis - the symta/s approach," in *IEEE Proceedings Computers and Digital Techniques*, 2005.

[42] S. Martin, *Maitrise de la dimension temporelle de la qulité de service dans les réseaux*. PhD thesis, Université Paris XII, 2004.

[43] S. Martin and P. Minet, "Schedulability analysis of flows scheduled with FIFO: application to the expedited forwarding class," in *Proceedings of the 20th international conference on Parallel and distributed processing*, IPDPS'06, (Washington, DC, USA), IEEE Computer Society, 2006.

[44] S. Martin and P. Minet, "Worst case end-to-end response times of flows scheduled with FP/FIFO," in *Proceedings of the International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies*, ICNICONSMCL '06, (Washington, DC, USA), pp. 54–, IEEE Computer Society, 2006.

[45] K. Tindell and J. Clark, "Holistic schedulability analysis for distributed hard real-time systems," *Microprocessing and Microprogramming - Parallel processing in embedded real-time systems*, vol. 40, pp. 117–134, April 1994.

[46] S. Martin, P. Minet, and L. George, "End-to-end response time with fixed priority scheduling: trajectory approach versus holistic approach," *International Journal on Communication Systems*, vol. 18, no. 1, pp. 37–56, 2005.

[47] S. Martin and P. Minet, "Improving the analysis of distributed non-preemptive FP/DP* with the trajectory approach.," *In Telecommunication Systems, Springer*, vol. 30, pp. 49–79, November 2005.

[48] H. Bauer, J.-L. Scharbarg, and C. Fraboul, "Improving the worst-case delay analysis of an AFDX network using an optimized trajectory approach," *IEEE transactions on industrial infromatics*, vol. 6, pp. 521–533, November 2010.

[49] X. Li, J.-L. Scharbarg, and C. Fraboul, "Analysis of the pessimism of the trajectory approach for upper bounding end-to-end delay of sporadic flows sharing a

switched ethernet network," in *19th International Conference on Real-time and Networked Systems*, (Nantes, France), 2011.

[50] H. Bauer, J.-L. Scharbarg, and C. Fraboul, "Applying and optimizing trajectory approach for performance evaluation of AFDX avionics network," in *14th IEEE conference on, Emerging Technologies and Factory Automation*, (Mallorca, Spain), pp. 1–8, 22-26 September 2009.