Mälardalen University Press Licentiate Theses
No. 184

MULTI-HOP REAL-TIME COMMUNICATION OVER SWITCHED ETHERNET TECHNOLOGY

Mohammad Ashjaei

2014

School of Innovation, Design and Engineering

ISBN 978-91-7485-168-7
ISSN 1651-9256
Printed by Arkitektkopia, Västerås, Sweden

Populärvetenskaplig sammanfattning

Trenden med att använda datorer för att underlätta vardagen i våra liv visar på en stor ökning av antalet datorer under de senaste decennierna. Vi använder oss av datorer överallt, allt ifrån som en del av våra dagliga rutiner i enklare uppgifter, till exempel i mobiltelefoner och hushållsapparater, till i mer komplicerade och sofistikerade uppgifter som återfinns i avancerade system i moderna bilar och som automation av industri och tillverkning. I motsats till klassiska datorer med tangentbord och skärm, utformade till att kunna göra en uppsjö av olika uppgifter, är dessa datorsystem konstruerade för att utföra specifika funktioner. Vi kallar dessa datorsystem för inbyggda system, där datorn är inbyggd i systemet och den gör en mer eller mindre specifik uppgift.

För att få en funktionalitet med hjälp av dessa system är själva funktionen i praktiken ofta fördelad över många olika inbyggda system, och de inbyggda systemen som ingår in den specifika funktionen är förbundna med varandra via ett kommunikationssystem. Vi kallar denna typ av system för distribuerade inbyggda system. Till exempel, en krockkudde i en modern bil innehåller flera sensorer som lämnar information till en styrenhet för att besluta om bilen har krockat och i så fall hur och när krockkudden ska blåsas upp. Informationen från sensorerna som registrerar en potentiell krock är ämnad till styrenheten som ska fatta beslut. Denna information skickas via meddelanden i det distribuerade inbyggda systemet.

Många av dessa distribuerade inbyggda system har specifika tidskrav som måste uppfyllas för att korrekt funktionalitet ska uppnås. Meddelanden som skickas över nätverket måste tas emot vid sina respektive destinationer inom en viss tid, som kallas deadline. Vi kallar denna typ av system för distribuerade inbyggda realtidssystem. Beroende på vilken typ av funktionalitet det är

frågan om så kan en miss av deadline, dvs att ett meddelande levereras för sent, leda till katastrofala konsekvenser. Till exempel så kan ett sent meddelande om krock av bilen leda till att krockkudden blåses upp för sent och därmed mister sin funktion. För att säkerställa tidsmässig leverans av meddelanden så definieras typiskt ett protokoll för att på ett förutsägbart sätt skicka och ta emot meddelanden. Detta protokoll används sen av alla inbyggda system sammankopplade med kommunikationssystemet, vilket leder till att systemen kan prata med varandra och förstå varandra.

I denna avhandling fokuserar vi på ett särskilt protokoll, baserat på Ethernetteknik. Protokollet ger tidsgaranti för leverans av meddelanden och lämpar sig där med för realtidsystem. Detta protokoll är FTT-SE-protokollet, som ursprungligen är utvecklat för relativt enkla nätverk bestående av en enda Ethernetswitch. I många industriella tillämpningar finns där tiotals inbyggda system som i och med sitt stora antal inte kan sammankopplas av endast en Ethernetswitch. Därför behövs en lösning för kommunikationssystem där flera Ethernetswitchar är sammankopplade, samt där garantier kan ges för den tidsmässiga leveransen av meddelanden som skickas över flera switchar. Det huvudsakliga målet med denna avhandling är att föreslå lösningar för att anpassa FTT-SE-protokollet för distribuerade inbyggda realtidssystem bestående av flera switchar. Vi presenterar två tekniska lösningar och vi presenterar tidsanalyser för alla föreslagna lösningarna så att tidsgarantier kan ges. Vidare utvärderar vi våra lösningar med hjälp av simulering så att de kan jämföras samt så att vi kan presentera en effektiv lösning för varje typ av system.

Abstract

Switched Ethernet technology has been introduced to be exploited in real-time communication systems promoted by its features such as high throughput and wide availability. Given the wide availability of switched Ethernet technology, it is also a cost-effective solution. In recent years many real-time switched Ethernet protocols have been developed. However, there is a challenge in preserving the profits of traditional Ethernet technology, and at the same time overcoming the limitations imposed by using Commercial Off The Shelf (COTS) switches. These limitations mainly originate from the non-deterministic behavior of such Ethernet switches inherent in the use of FIFO queues for internal message buffering, and a limited number of priority levels available to differentiate messages.

In our research we focus on two particular solutions to provide real-time communication using switched Ethernet technology, one based on COTS Ethernet switches named the FTT-SE architecture and the other using a modified Ethernet switch called the HaRTES architecture. Both architectures are based on a master-slave technique and they support different and temporally isolated traffic types including synchronous and asynchronous traffic. Also, they provide mechanisms implementing adaptivity as a response to the requirements imposed by dynamic real-time applications. Nevertheless, the two mentioned architectures were originally developed for a simple network consisting of a single switch, and they were lacking support for multi-hop communication. In industrial applications, multi-hop communication is essential as the networks comprise a high number of nodes that is far beyond the capability of a single switch.

In this thesis, we study the challenges of providing multi-hop communication using the FTT-SE and HaRTES architectures. We propose different architectures to provide multi-hop communication while preserving the key characteristics of the original single-switch architecture, such as timeliness guar-

antee, resource efficiency, adaptivity and dynamicity. We develop a response time analysis for each proposed architecture and we compare them to assess their corresponding benefits and limitations. Further, we develop a simulation tool to evaluate the solutions.

To Sara, My Soulmate!

Acknowledgments

First of all, I take this opportunity to express my profound gratitude to my supervisors Prof. Thomas Nolte and Dr. Moris Behnam for their constant encouragement throughout my studies. They have been guiding me from my master thesis with an exemplary support, useful suggestions, comments and feedback. Thomas has always influenced me by his great positive attitude bringing to work. Also, I appreciate the valuable feedback of Moris which have improved my work. It may sound cliché, but without your support and help this thesis would not be possible.

Next, I would like to express a deep sense of gratitude to my co-authors. I am very thankful to Prof. Luis Almeida (University of Porto) and Prof. Paulo Pedreiras (University of Aveiro) for their cordial support and valuable information. I would also like to thank Prof. Reinder J.Bril (University of Eindhoven) for his precious information during the meetings and discussions. A big thank you goes to Dr. Guillermo Rodriguez-Navas not just due to discussions on ”Synchronization”, but also because of many discussions on music, cinema and theater. I would also like to thank Dr. Saad Mubeen for his valuable feedback on the thesis.

Further, I wish to express my appreciation to the lecturers and professors who I learned a lot from during meetings, lectures, seminars and PhD courses. My exceptional thank you goes to Damir Isovic (who was my first teacher and mentor at MDH), Hans Hansson, Ivica Crnkovic, Mikael Sjödin, Thomas Nolte, Paul Pettersson, Emma Nehrenheim, Monica Odlare, Jan Gustafsson, Björn Lisper, Cristina Seceleanu, Jan Carlson, Radu Dorbin, Dag Nyström, Ning Xiong, Kristina Lundqvist, Harold (Bud) Lawson, Giacomo Spampinato, Gordana Dodig-Crnkovic, Lars Asplund, Mats Björkman, Mikael Ekström, Moris Behnam, Maria Lindén, Baran Çürüklü and Jukka Mäki-Turja. I also would like to appreciate IDT administration staff for their help with practical issues. Many thanks Carola, Susanne, Sofia, Ingrid, and the others.

During my studies I had a chance to visit University of Porto and University of Aveiro in Portugal as a visiting researcher for one month. I had a great time there in a very beautiful September. I gained several nice experiences, visiting small and big cities, tasting delicious cookies and Port wines. I got to know many colleagues working on different areas. I had a lot of great chats and discussions with them. In particular, I would like to thank Paulo Pedreiras and his family, Luis Almeida, Ricardo Marau, Luis Silva and Milton Cunguara. You made my visit full of joy and work, Obrigado!

I am obliged to my friends and colleagues at the department for all the fun we had during these years in conference trips, fika, parties, bars and badminton. I wish to thank my office-mates Svetlana, Saad and Rafia; PhD-students (and former members) in my research group Sara Af., Nima, Meng, Matthias, Hamid, Daniel, Hang, Mikael, Farhang; and in an alphabetic order Abhilash, Adnan, Aida, Alessio, Andreas G., Aneta, Anna, Antonio, Apala, Arash, Batu, Cristina, Dag, Daniel K., Daniel S., Eddie, Elena, Federico, Francisco, Fredrik Ek., Gabriel, Gregory, Guillermo, Hus, Husni, Irfan, Ivan, Jagadish, Jiale, Josip, Juraj, Kan, Kivanc, Leo, Luka, Mahnaz (Anita), Mehrdad, Melika, Martin, Nesredin, Nikola, Omar, Pablo, Per, Predrag, Raluca, Sara Ab., Sara Ab. Assadollah, Sara D., Séverine, Simin, Shahina, Stephan (Bob), Tibi, Wasif, Yue and others.

I would also like to thank my Iranian friends in Västerås who have been always supportive and made life out of my home country easier. A big salute goes to Mersedeh, Siavash, Nima, Amir, and the others.

Last but not least, I would like to take this opportunity to thank my family. In particular, my parents for their endless love, tremendous support and encouragement during my entire life. I am distinctly thankful to my wife Sara, who was always patient and supportive during all up and down in our life. I am a very lucky man to have you! Also, I am grateful of my little siblings, Saeedeh and Soheil, whom together we could laugh for few hours constantly without being drunk!

This work has been supported by the Swedish Foundation for Strategic Research under the project PRESS.

Mohammad Ashjaei
17:43, October 24th, 2014
Västerås, Sweden

List of publications

Papers included in the licentiate thesis[1]

Paper A *Performance Analysis of Master-Slave Switched Ethernet Network*. Mohammad Ashjaei, Moris Behnam, Luis Almeida, Thomas Nolte. In the Proceedings of the 8th IEEE International Symposium on Industrial Embedded Systems (SIES), 2013, May.

Paper B *Response Time Analysis of Multi-Hop HaRTES Ethernet Switch Networks*, Mohammad Ashjaei, Paulo Pedreiras, Moris Behnam, Reinder J. Bril, Luis Almeida, Thomas Nolte. In the Proceedings of the 10th IEEE International Workshop on Factory Communication Systems (WFCS), 2014, May.

Paper C *Reduced Buffering Solution for Multi-Hop HaRTES Switched Ethernet Networks*, Mohammad Ashjaei, Moris Behnam, Paulo Pedreiras, Reinder J.Bril, Luis Almeida, Thomas Nolte. In the Proceedings of the 20th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA), 2014, August.

Paper D *Implementing a Clock Synchronization Protocol on a Multi-Master Switched Ethernet Network*, Mohammad Ashjaei, Moris Behnam, Guillermo Rodriguez-Navas, Thomas Nolte. In the Proceedings of the 18th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), 2013, September.

[1] The included articles have been reformatted to comply with the licentiate thesis layout.

Paper E *SEtSim: A Modular Simulation Tool for Switched Ethernet Networks*, Mohammad Ashjaei, Moris Behnam, Thomas Nolte. MRTC report, ISSN 1404-3041, Mälardalen Real-Time Research Centre, Mälardalen University, September, 2014 (Under Review in a Journal).

Additional papers, not included in the licentiate thesis

1. *Evaluation of Dynamic Reconfiguration Architecture in Multi-Hop Switched Ethernet Networks*, Mohammad Ashjaei, Paulo Pedreiras, Moris Behnam, Luis Almeida, Thomas Nolte, In the Proceeding of the 19th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), Work-in-Progress Session, 2014, September.

2. *Dynamic Reconfiguration in Multi-Hop Switched Ethernet Networks*, Mohammad Ashjaei, Paulo Pedreiras, Moris Behnam, Luis Almeida, Thomas Nolte, In the Proceeding of the 6th Workshop on Adaptive and Reconfigurable Embedded Systems (APRES), 2014, April.

3. *Supporting Multi-Hop Communications with HaRTES Ethernet Switches*, Mohammad Ashjaei, Paulo Pedreiras, Moris Behnam, Luis Almeida, Thomas Nolte, In the Proceeding of the IEEE Real-Time Systems Symposium (RTSS), Work-in-Progress Session, 2013, December.

4. *MTU Assignment in a Master-Slave Switched Ethernet Network*, Mohammad Ashjaei, Moris Behnam, Luis Almeida, Thomas Nolte, In the Proceeding of the 12th International Workshop on Real-Time Networks (RTN), 2013, July.

5. *Worst-Case Delay Analysis of Master-Slave Switched Ethernet Networks*, Mohammad Ashjaei, Meng Liu, Moris Behnam, Ahlem Mifdaoui, Luis Almeida, Thomas Nolte, In the Proceeding of the 2nd International Workshop on Worst-Case Traversal Time (WCTT), 2012, December.

6. *The Design and Implementation of a Simulator for Switched Ethernet Networks*, Mohammad Ashjaei, Moris Behnam, Thomas Nolte, In the Proceeding of the 3rd International Workshop on Analysis Tools and Methodologies for Embedded and Real-Time Systems (WATERS), 2012, July.

7. *A Compact Approach to Clustered Master-Slave Ethernet Networks*, Mohammad Ashjaei, Moris Behnam, Thomas Nolte, Luis Almeida, Ricardo Marau, In the Proceeding of the 9th International Workshop on Factory Communication Systems (WFCS), Work-in-Progress Session, 2012, May.

8. *Support for Hierarchical Scheduling in FreeRTOS*, Rafia Inam, Jukka Mäki-Turja, Mikael Sjödin, Mohammad Ashjaei, Sara Afshar, In the Proceeding of the 16th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), 2011, August.

Contents

I

Thesis

Chapter 1

Introduction

Over the last decades, the communication requirements of networked real-time embedded systems have become overly complex such that the conventional communication protocols have shown to be impotent. The complexity arises from advances in embedded equipments and increments in their functionalities along with an increasing massive amount of information to be exchanged within the embedded systems. Besides this complexity, many other challenges are imposed by new requirements in the real-time networks. These new requirements include incorporating the traffic with diverse activation patterns (event-triggered and time-triggered) and Quality-of-Service (QoS) management allowing for on-the-fly reconfiguration. For instance, many industrial media control applications [1] such as machine vision [2], automated inspection [3], vehicle guidance [4] and in-vehicle networks [5] [6], have requirements both on hard real-time and media processing quality. Therefore, the network resources used by the multimedia components must be predictable, both in terms of how and when. Such predictability can be achieved by a QoS management.

In order to cope with the mentioned complexity, switched Ethernet has been introduced with good properties such as high throughput, expandability and being a cost effective solution for networked embedded systems. Compared to other network technologies used in embedded real-time systems, Ethernet can support a higher throughput, up to 100Mbps [1]. Moreover, it is expandable and flexible in the sense that the switches can be connected together in different topologies to build a multi-hop architecture. Due to the wide availability of

[1] Using 1Gbps in industrial applications has not been investigated thoroughly yet.

Ethernet switches, using them in industrial applications is also cost-effective.

However, the non-deterministic behavior of Commercial Off The Shelf (COTS) Ethernet switches impaired its use in time-critical applications. Basically, the queues in the switch ports may overflow due to uncontrolled arrival of packets, a situation that, in a worst-case, leads to drop of packets. Moreover, COTS switches typically have First In First Out (FIFO) queues that can generate long blocking times for urgent packets. This latter problem can be partially mitigated prioritizing the traffic and using separate queues for different priorities. Nevertheless, the IEEE 802.1D standard considers up to 8 priority levels, which is too few to support effective priority scheduling. Therefore, many Real-Time Ethernet (RTE) protocols have been introduced to overcome these limitations, while they profit from the good features of traditional Ethernet technology.

The RTE protocols are categorized into two main groups. The first group utilizes enhanced Ethernet switches. For instance, TTEthernet [7] [8] and PROFINET IRT [9] were proposed and optimized for time-triggered operation. As a second group of RTE protocols, several solutions were researched and eventually marketed based on overlay protocols that control the traffic submitted to the COTS switches. For this group we can mention Ethernet POWERLINK [10] which uses a master-slave technique. However, it turned out that most of the RTE protocols found in the literature have severe limitations in dealing with dynamic real-time applications. These applications are characterized by having evolving requirements (e.g. message streams may be added, removed and updated during run-time) which, despite being volatile, are subject to strict timeliness requirements. RTE protocols that provide strict determinism typically adopt static scheduling, thus impeding any sort of effective online adaptation to the communication requirements.

As a result, two solutions have been developed in order to deliver adaptivity, dynamicity and effective distribution of resources in real-time networked embedded systems. The first solution, called Flexible Time-Triggered Switched Ethernet (FTT-SE) [11], was developed over COTS Ethernet switches, while the second solution, called Hard Real-Time Ethernet Switching architecture (HaRTES) [12] [13], is a solution using enhanced Ethernet switches. Both solutions are based on the master-slave technique and they were originally proposed for a simple network consisting of a single switch. The FTT-SE and the HaRTES architectures support real-time periodic, real-time sporadic and non-real-time traffic. Further, a middleware was developed [14] that uses a linear time-complexity online admission control [15] in order to provide dynamic reconfiguration. In this thesis we focus on these two architectures.

1.1 Goal of the Thesis

The FTT-SE and HaRTES architectures provide real-time communication services based on the Flexible Time Triggered (FTT) paradigm [16]. Further, they cater adaptive reconfiguration to the response of dynamic real-time applications. Both architectures use a master-slave technique in which a master module controls transmission of the traffic through the switch. In the FTT-SE architecture, the master module is implemented in a particular node attached to the switch and it triggers the transmission from the slave nodes. This architecture presents a low cost solution compared to the solutions that use enhanced Ethernet switches. This is due to the wide availability of the COTS Ethernet switches, also at low prices. Despite the mentioned profit of the FTT-SE architecture, it presents some structural limitations. For instance, all nodes need to be FTT-compliant which in turn requires a specific network device driver to be used in the operating system of the connected nodes. On the other hand, the HaRTES architecture overcomes these limitations by accommodating the master module inside the switch.

Both the FTT-SE and HaRTES architectures were originally developed for a simple network consisting of a single switch. However, the networked embedded systems in industrial applications usually comprise a high number of nodes, which is far beyond the capability of a single switch. Therefore, the multi-hop communication is necessary for such applications. The main goal of this thesis is: *to build a multi-hop communication using the FTT-SE architecture as well as the HaRTES architecture, while preserving the features of the single-switch case including timeliness guarantee, efficient usage of resources and adaptivity.*

1.2 Research Challenges

Connecting the switches together in order to build a multi-hop communication using the FTT-SE and HaRTES architectures arises many challenges. In this section we describe these challenges.

For each of the FTT-SE and HaRTES architectures, a protocol has been presented to transmit messages between nodes that are connected to a single switch. These two protocols have few differences in handling sporadic messages. In both protocols the master module is responsible to schedule the messages in fixed-duration time-slots. However, when extending the architectures to multiple switches, the existing protocols are not sufficient anymore. Ba-

sically, the messages may not merely cross through one switch, instead the messages may traverse several switches to reach the destination node. This introduces several challenges regarding dealing with the mentioned situation.

Firstly, the scheduler in the master module requires to take into account all switching delays and interference that may occur to a message crossing the switches in order to schedule the message. Also, the scheduler needs to have an overall view on the message transmission in all links between the switches and the nodes such that it can schedule a message passing over the links. For instance, the scheduler needs to know how much bandwidth is available in each link for scheduling a particular message. Secondly, in case of the multi-hop FTT-SE architecture, considering one master module to perform the scheduling process may not be efficient as shown in [17]. Thus, it may be worth to have several master modules to organize the message transmission. In case of the multi-hop HaRTES architecture, having multiple master modules is inevitable as they are residing inside the HaRTES switches. Therefore, in both cases, the challenge is to make consistent schedulers in the master modules in order to build a collaboration among them for scheduling and forwarding the traffic. Moreover, this consistency among the master modules, hence the schedulers, requires a time synchronization protocol with a small overhead on the traffic transmission and bandwidth. Thus, another challenge is to provide such a time synchronization protocol. Note that the FTT-SE and HaRTES architectures have some fundamental differences in the implementation of the master module. Therefore, providing different protocols for the FTT-SE and HaRTES architectures may be more suitable.

Considering the above outlined challenges, the existing protocols for the FTT-SE and HaRTES architectures should be extended to support multi-hop communication. The extended protocols require to cover scheduling method, traffic transmission through multiple switches, and synchronization among master modules. Note that the extended protocols have to preserve the properties of the existing protocols such as timeliness guarantee and resource efficiency.

For the sake of timeliness guarantee, a response time analysis [18] for the traffic that is transmitted within multi-hop architectures is demanded. Moreover, one key feature of the architectures is adaptiveness which is the capability to add, remove or change components dynamically (e.g., add a new message during run-time without disruption of the guarantees already provided to the existing set of messages). An admission control is required to accept the changes in the system. The admission control issues the decisions based on the response time analysis. In order to capture the behavior of the extended protocols, different corresponding response time analysis methods may be required.

Thus, the challenge is to provide a suitable response time analysis for the two architectures while at the same time achieving a low pessimism level of the respective analyses.

The extended protocols for the multi-hop architectures need to be evaluated from different perspectives to draw conclusions on their respective performance. This evaluation includes the validation of the response time analysis and evaluation of the protocols' performance with respect to traffic latency and bandwidth utilization. The challenge is to provide such an evaluation using methods like simulation, experiments or mathematical proofs.

The above stated challenges can be formulated into the following research challenges.

1. Extend the existing protocols for the multi-hop FTT-SE architecture as well as for the multi-hop HaRTES architecture such that they are able to schedule and forward traffic across multiple switches in the respective multi-hop architectures. The extended protocols should provide timeliness guarantee while at the same time comply with an efficient usage of resources.

2. Develop an efficient response time analysis for each of the extended protocols. The analyses should be able to capture the behavior of the extended protocols in their respective worst-case behavior, and at the same time they should exhibit a low level of pessimism.

3. Evaluate the performance of the extended protocols with respect to traffic latency and bandwidth utilization.

1.3 Technical Contributions

In this section we present the contributions of the thesis that address the formulated research challenges. To provide a resolution for multi-hop communication using the FTT-SE and HaRTES architectures we propose for each architecture solutions containing a transmission protocol, a scheduling algorithm and a response time analysis. We perform a comparative evaluation of the respective solutions. Based on the evaluation, we select the most effective one with respect to resource utilization and traffic response times.

We organized the contributions into six parts. Contribution 1 and 2 propose two solutions for the multi-hop FTT-SE architecture. Then, Contribution

3 and 4 propose two solutions for the multi-hop HaRTES architecture. Contribution 5 proposes a synchronization method that performs better than previously proposed techniques. Finally, Contribution 6 presents a simulation tool for evaluation of the extended protocols.

Personal contribution. The research presented in this thesis is done in collaboration with University of Porto, University of Aveiro, University of Eindhoven and University of Illes Balears. I am the main contributor and the first author of all included papers. Prof. Thomas Nolte and Dr. Moris Behnam are my supervisors and contributed in reviewing the solutions and discussions. Prof. Luis Almeida, Prof. Paulo Pedreiras and Prof. Reinder J.Bril contributed in reviewing and discussions on the solutions and analyses. Also, Dr. Guillermo Rodriguez-Navas contributed in discussions on the clock synchronization protocol and its evaluation.

1.3.1 Contribution 1: The Multi-Master FTT-SE Architecture

In the context of the original FTT-SE architecture, we propose to connect multiple switches in a tree topology and to connect a master node to each switch in order to control the traffic transmission in that switch. We also provide a response time analysis for the traffic transmitted through the switches in this architecture. Finally, we validate the analysis using network examples. We also evaluate the multi-master architecture with a simulation technique. The details of this contribution are presented in Paper A.

1.3.2 Contribution 2: The Hybrid FTT-SE Architecture

In order to improve the previous solution (i.e., the multi-master FTT-SE architecture) with respect to the resource utilization, we propose another solution to forward the traffic in the multi-hop FTT-SE architecture. We again consider an architecture of multiple switches connected in a tree topology. However, instead of using one master node per switch, we propose to assign one master node per group of switches. This group of switches is identified by a number of switches that share a parent switch and it is called a cluster. Thus, each master node is responsible to coordinate the traffic of its cluster. We present a response time analysis for this architecture and compare that with the multi-master architecture in terms of bandwidth utilization. We show that the hybrid FTT-SE architecture performs better with respect to bandwidth utilization compared to

the multi-master and the single-master architectures [19]. Contribution 2 is described in Paper A.

1.3.3 Contribution 3: The Distributed Global Scheduling

In the context of the HaRTES architecture, we propose to connect multiple HaRTES switches in a tree topology and we introduce a method called Distributed Global Scheduling (DGS) to forward the traffic through the switches. We present a response time analysis and verify the analysis using a network example. We also investigate the applicability of the architecture on an industrial automotive case study. This contribution is presented in Paper B.

1.3.4 Contribution 4: The Reduced Buffering Scheme

As the DGS method is not easy to expand for large networks, we propose another method which is called Reduced Buffering Scheme (RBS). In contrast to the DGS method, the RBS method can cater lower latency for the traffic. We propose a response time analysis for this method and compare it with the DGS method using their response time analysis. We show that in most of the cases the RBS method can deliver the traffic much faster than the DGS method. This contribution is presented in Paper C.

1.3.5 Contribution 5: The Clock Synchronization Method

In all proposed solutions for the multi-hop FTT-SE and the multi-hop HaRTES architectures, we propose a synchronization technique between the master modules using a signaling method. However, to achieve better performance, we adapt the FTT-SE architecture to support a clock synchronization standardized as IEEE 1588. We study the effects of applying this standard on the multi-hop FTT-SE architecture and verify that solution using a model checker called Uppaal [20]. Although, the synchronization protocol is investigated on the FTT-SE architecture, it can be directly used in the HaRTES architecture as they have similar techniques used for synchronization, i.e., they both use the master-slave technique in the context of the FTT paradigm. This contribution is presented in Paper D.

1.3.6 Contribution 6: The Simulation Tool (SEtSim)

In order to evaluate the performance of different extended protocols for multi-hop architectures, experiments or simulations are required. However, performing experiments for large scale network topologies, potentially with complex message sets, is not straightforward. In particular when the protocols are in their early stages of design. Therefore, in this thesis, we use a simulation-based approach that is more time efficient compared to performing experiments on hardware. In order to conduct the simulation, we develop a simulation tool based on Simulink. The tool is modular and it can be easily changed to support different scheduling policies. Also, it is able to accommodate other Ethernet-based protocols. We use this tool to evaluate the extended protocols and to validate the response time analyses. This contribution is presented in Paper E.

The relation between the research challenges, the contributions and the publications is illustrated in Table 1.1.

	Challenge 1	Challenge 2	Challenge 3	Publications
Contribution 1	√	√	√	Paper A
Contribution 2	√	√	√	Paper A
Contribution 3	√	√	√	Paper B
Contribution 4	√	√	√	Paper C
Contribution 5	√			Paper D
Contribution 6			√	Paper E

Table 1.1: The relation between the research challenges, the contributions and the publications

1.4 Research Method

In order to achieve the research goal, we adopted and followed a deductive-like research methodology. In the deductive research method a hypothesis based on an existing theory is developed. Then, a strategy to test the hypothesis is followed to reach a confirmation or a rejection [21]. This method of research, which is also called top-down approach, can be translated to define a main research goal and to make a strategy to reach that, as we have used the same in this thesis.

First we defined the main research goal by reviewing the state of the art. Then, we identified the research challenges based on the main research goal.

We proposed a solution for the identified challenges by reviewing the related literature.

We validated the proposed solution using a simulation technique and compared that with the existing or previously proposed solutions. In case the solution was not desirable, we tuned or completely altered the solution. After achieving a desirable solution we finalized that by publishing scientific reports or papers.

As the proposed solutions were not optimum considering the resource usage, cost and complexity of implementation, we studied their limitations. Focusing on the limitations we proposed new solutions for the identified challenges by again reviewing the related literature.

The flow of our research process is further illustrated in Figure 1.1.

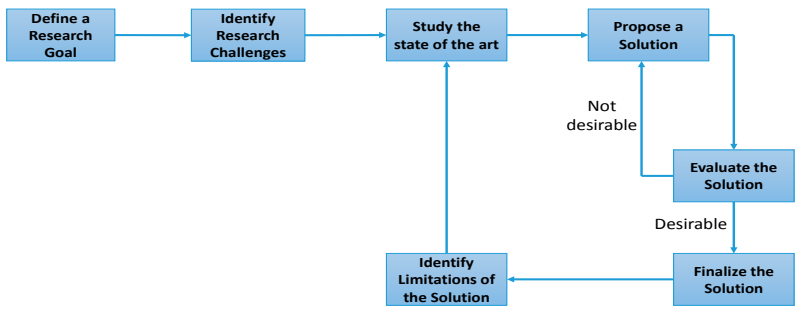

Figure 1.1: The Flow of Research Process

1.5 Outline of the Thesis

This thesis consists of 9 chapters and the outline of the rest of the thesis is organized as follows. Chapter 2 describes some required backgrounds on the thesis. Chapter 3 presents the basics of the FTT-SE and HaRTES architectures. Chapter 4 summarizes the thesis and presents the on-going work. Then, the included papers are presented in Chapter 5 to 9.

Chapter 2

Background

This chapter introduces some basic concepts about real-time systems and communications, in particular switched Ethernet technology, that are relevant to this thesis.

2.1 Real-Time Embedded Systems

An *embedded system* is a microprocessor-based system that is designed to perform a specific function [22]. They are mostly components of a larger system, where they often interact with the environment through sensors and actuators. The embedded systems are found in almost all electronic devices ranging from simple items, such as home appliances and alarm clocks, to more sophisticated items, like modern cars and industrial control systems. Besides the high number of embedded systems that has sharply increased in the last decade, the software that runs on the embedded systems has increased drastically in both size and complexity. For instance, around 100 million lines of code are running in several embedded systems to get a premium car out on the driveway [23].

For some of the embedded systems, not only the logical correctness of the functionality is important, but also they have to produce the logical results within a specified time (i.e., a deadline). This type of embedded systems is called *real-time* embedded system. Depending upon the consequences that may occur because of a missed deadline, a real-time system can be categorized into two groups: Hard and Soft. If the results of an embedded system are produced after its specified deadline and that causes system failure, the system is called

a hard real-time system. Whereas, if the consequences of missing a deadline cause a performance degradation but still it has some utility for the system, the system is called a soft real-time system. For example, the airbag control system in a car is a hard real-time system, while video and audio streaming typically is denoted as soft real-time systems.

2.2 Real-Time Communications

Many embedded applications are implemented using a distributed architecture where the functionality is distributed over many embedded systems (nodes) interconnected by a communication system (network). Such systems are commonly known as distributed embedded systems, where the processing nodes require to exchange information. In case of real-time distributed systems, the temporal behavior depends on several items including timing requirements in both processing software in the nodes and messages exchanged within the network. In such a distributed system, the communication system has to deliver messages within specific timing constraints, which is known as a *real-time communication* system.

A good example is a modern car where many software functions, such as cruise control and anti-lock braking, are distributed among some of the 70 to 100 Electronic Control Units (ECUs) that are connected via different networks [24]. Looking specifically at the cruise control mechanism, there are several sensors needed to measure speed, throttle value and brake pedal position at the vehicle. The messages from these sensors are sent to a central computing unit via a communication system. A control loop in the unit provides a control value to be submitted to the actuators such as the throttle and the brake. The success of the system depends on the time it takes from the sensor measurement to the actuation.

2.2.1 Event- and Time-Triggered Communication

From the temporal behavior point of view, communication services can be distinguished to two different paradigms: *event-triggered* and *time-triggered* communication [25].

In time-triggered communication, all nodes have a common time reference established using a time synchronization method. The applications send messages at certain predefined points in time in a periodic manner. This paradigm can provide a great determinism as the schedule of message transmission can

be built in advance, thus it prevents simultaneous transmission which may lead to collision.

Based on the event-triggered communication paradigm, messages are sent by the node upon the occurrence of an event, such as termination of a task or an interrupt signal. These events may occur at any time instants, however, to define a worst-case scenario a minimum inter-arrival time is defined between consecutive events.

Each paradigm has some advantages which make them suitable for specific application. In general, the time-triggered approaches are suitable for control systems as they deal with periodic sampling and actuation, hence periodic messages. In contrast, the event-triggered approaches are more proper to handle sporadic messages like a sensor changing value in automation. In some applications both periodic and sporadic activities are occurring, thus a combination of the event-triggered and the time-triggered paradigm is desirable in order to gain from their respective profits. A usual technique to conduct this situation is to use communication slots containing two consecutive phases dedicated to each type of transmission. The size of the phases can be selected according to the need of a particular application. This way, each type of traffic is forced to respect the associated phase for transmission, hence a temporal isolation among them is guaranteed. This method has been used in many communication protocols such as FlexRay [26], FTT-CAN [27] and FTT-Ethernet [28].

In this thesis, we focus on two particular architectures, the multi-hop FTT-SE architecture [11] and the multi-hop HaRTES architecture [12], which handle the time-triggered and event-triggered paradigms.

2.2.2 Message Scheduling

Commonly, distributed systems are built over a shared medium network where the nodes exchange data messages. In order to maintain the timing requirements, access to the communication network should be properly scheduled. Therefore, scheduling of messages is required. The scheduling of messages can be done using the scheduling policies introduced in the microprocessor domain, however depending on the system requirements along with the hardware and software support. These scheduling policies include offline scheduling, and online scheduling with fixed or dynamic priority assignments [25].

Despite similarities of the network domain and the microprocessor domain, such as using scheduling policies and message activation patterns (periodic, sporadic or aperiodic), there are additional challenges in real-time distributed systems regarding the scheduling process. These challenges are imposed by

the nature of the distributed architecture.

One key element in scheduling is to keep track of available resources, which is not always easy in distributed systems. In fact, the knowledge of the system state, in particular network and nodes, is not always available. This makes the scheduling decision to be forced based on incomplete information [29].

Moreover, in the context of scheduling in microprocessors, preemptive scheduling is commonly used as, in average, it provides a higher schedulability ratio compared to non-preemptive scheduling [30]. However, in communication systems a packet cannot be preempted during its transmission. Packets are the result of a message fragmentation so preemption can be applied at packet level.

2.3 Switched Ethernet Technology

Ethernet was originally developed as a technology for computer networking. Nowadays, Ethernet is supported by a collection of IEEE standards defining different network layers. Besides evolution in the transmission speed, Ethernet evolved in the physical layer from a bus topology to a star and switching hubs (commonly known as Ethernet switches) topology.

Ethernet uses a shared medium for communication, thus it creates a single collision domain. This means that a message is transmitted in the medium and all nodes can receive that message. Also, an arbitration mechanism called Carrier Sense Multiple Access with Collision Detection (CSMA/CD) is used to overcome the collision problem. Essentially, a node holds the message until the shared medium becomes idle. As all nodes are following the same rule, the collision becomes inevitable. In case of collision all of the transmissions involved in the collision are aborted and the nodes wait for a random amount of time, then they start retransmission of the messages. This procedure continues until a successful transmission, i.e., without collision. This problem of message retransmission, which leads to a low efficiency in using the bandwidth, was the main motivation to use Ethernet switches.

A switch provides a private collision domain for each of its ports, hence it reduces the impact of the non-deterministic feature of the original Ethernet inherent in the CSMA/CD arbitration. This section describes the basics of switched Ethernet followed by protocols that have been developed to use the switches in real-time applications.

2.3.1 Switched Ethernet

Most common switches are based on IEEE 802.1D that introduces FIFO queues for each output port. A message arrives to a switch and after recognizing the destination address, it is buffered in the output queues. The message stays in the queue for the time needed to transmit the messages ahead in the queue. According to the IEEE 802.1D standard, for each output port a limited number of parallel queues for different priority levels are considered. Messages that are queued in different priority levels are distributed based on their priorities starting from the higher priority queue. The basic structure of an Ethernet switch is depicted in Figure 2.1.

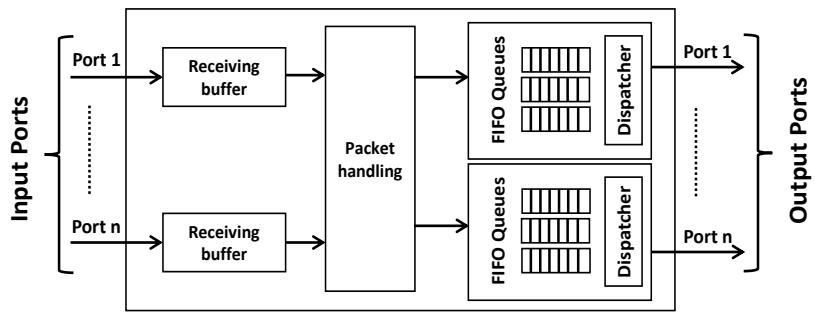

Figure 2.1: The Switch Internal Structure

Ethernet switches commonly support three types of message forwarding: unicast, broadcast and multicast. In unicast forwarding a message is transmitted from one source node to one destination node, directly, using the destination MAC address. Whereas, the broadcast forwarding type transmits a message from one source node to all other nodes connected to the switch simultaneously. This way the MAC address should be filled with the MAC broadcast address so that the switch can recognize the message to be sent to all output ports. Moreover, the multicast forwarding is transmitting a message from one source node to a group of nodes connected to the switch. The packet handling module depicted in Figure 2.1 performs message forwarding based on the MAC addresses.

Moreover, Ethernet switches can be either store-and-forward or cut-through type. A store-and-forward switch receives a message and entirely buffers that

message before forwarding it to the related output port. Whereas, the cut-through switch starts forwarding a message before it completely receives the message. In the store-and-forward method, the propagation of erroneous messages can be prevented by entirely buffering the message and checking that the message is correct.

2.3.2 Real-Time Protocols over Switched Ethernet

Using COTS switches in real-time systems may affect the possibility of guaranteeing timeliness behavior. The queues in the switch may overflow due to uncontrolled arrival of messages, which leads to messages being dropped. Also, due to the FIFO queues, urgent messages may be blocked for a long time. Therefore, many solutions are studied to overcome these limitations. The literature on switched Ethernet is vast and there have been many works addressing its adequacy to real-time communication. Herein, we discuss some of those protocols.

EDF Scheduled Switching

The EDF scheduled switching [31] is based on channel reservation mechanisms supported on enhanced switches. The real-time traffic is scheduled according to the Earliest Deadline First (EDF) scheduling policy. The system architecture contains a real-time layer on the switches and end nodes. This layer performs required tasks including real-time connections establishment, admission control, time synchronization and forwarding of the real-time traffic.

In order to transmit the real-time traffic, a real-time channel should be established. The channel establishment is done using a request and answer mechanism. The node, which has a real-time message ready to be sent, issues a request to the switch containing the message information such as its period, deadline and transmission time. The switch checks the feasibility of transmitting the message using an admission control and it sends the request to the destination node. The destination node also analyzes the message and sends back the answer to the switch. The switch forwards the answer back to the sender node. The answer could be success, in which case the message is transmitted, or could be reject.

EtheReal

Similar to the EDF scheduled switching, EtheReal [32] is based on channel reservation along with enhanced Ethernet switches. This protocol supports real-time and non-real-time traffic. Two services, one to support the real-time traffic and another for non-real-time traffic, are implemented in the switch. In this proposal there is no need to modify the operating system in the end nodes. The real-time service provides a reserved bandwidth by means of a connection setup to send the traffic with a minimum delay. The non-real-time service is developed specifically for legacy traffic without providing guarantees.

The connection setup is done by sending a reservation request to a user-level process called Real-Time Communication Daemon (RTCD). The reservation request contains the respective Quality of Service requirements including average traffic rate and maximum burst length.

TTEthernet

TTEthernet [33] [8] is a protocol based on an enhanced Ethernet switch and it is optimized for time-triggered transmission. The protocol supports event-triggered (ET) traffic and time-triggered (TT) traffic. The TTEthernet switch handles the former type according to the store-and-forward paradigm without any guarantee. However, the latter type is coordinated by a scheduler implemented in the TTEthernet switch and is transmitted according to the cut-through paradigm with a pre-defined delay guarantee.

In case of conflict between ET and TT traffic, the switch preempts the ET traffic and forwards the TT traffic with the guaranteed delay. The ET traffic is then retransmitted after termination of the TT traffic transmission. The TT Ethernet frame in this proposal contains a specific type field value ($0x88d7$) in order to inform the switch about its type.

PROFINET IRT

PROFINET IRT [34] [35] [9] is developed based on modified Ethernet switches and it is part of the IEC 61158 standard. The protocol supports different classes of services according to the applications. These classes include RT class UDP, RT class 1 to RT class 3 and non-real-time (NRT) class which has lower priority than RT class UDP. RT class 3 has the highest priority and it is forwarded according to a static communication schedule. The communication is cyclic-based and each cycle is divided into individual communication intervals. The

individual intervals are reserved for the classes and each node must respect that using a precise clock synchronization.

The traffic transmission is planned offline and loaded into the nodes. Other transmissions besides the planned ones are not allowed and in case of having those the switches block them.

AFDX

Avionics Full DupleX Switched Ethernet (AFDX) [36] is developed as a network communication specification with enhanced forwarding. AFDX has been used mainly in avionics. According to the protocol, a Virtual Link (VL) is defined as a logical communication channel between one source node and one or more destination node. Each VL is characterized by a Bandwidth Allocation Gap (BAG) which is a minimum time interval between consecutive frames. In order to enforce this characteristic in the source nodes a traffic shaper is developed to provide a deterministic communication behavior.

Offline planning for the traffic transmission is required to enforce the end-to-end delay guarantees which can be provided using an end-to-end delay analysis (e.g., Network Calculus [37] or Trajectory Approach [38]). Moreover, the AFDX switches contain forwarding policies based on the VL concepts which are statically defined. This feature eliminates online reconfiguration in the AFDX architectures.

Ethernet AVB

More recently, Audio Video Bridging (AVB) [39], as a set of technical standards developed by IEEE, is gaining a momentum, and it is mainly designed to be used in the automotive industry. Ethernet AVB supports clock synchronization, bandwidth reservation and traffic shaping services. Nevertheless, this protocol has some intrinsic limitations, such as a low number of priorities (maximum 8) and lack of explicit support for synchronous traffic. These limitations restrict the capabilities of the protocol in terms of specification of the stream properties.

In the AVB standards all messages must have priorities and messages can share a priority level. A set of traffic with the same priority belong to a same traffic class. Also, within each traffic class the messages are treated following the FIFO policy. Two traffic classes, known as class A and B, are defined in the standard, where class A has higher priority than class B. A Credit-Based Shaping Algorithm (CBSA) for each of these two classes is specified to handle

the traffic in class A and B. The traffic in class A (similarly class B) can be forwarded when the credit is zero or positive. During the transmission the credit is reduced with a defined rate called *sendSlope*. The credit is replenished at a particular rate which is called *idleSlope*. The credit is replenished in two scenarios. First, if there is no message in the corresponding traffic class in the queue, and second, when a message is waiting for transmission and there is another interfering message being transmitted. Note that, when the credit is positive and there is no message in the queue to forward, the credit becomes zero immediately. The non-real-time traffic is transmitted when there is no message in class A and class B waiting, or when the credit for them is negative.

There are some variations of the Ethernet AVB being proposed to provide a support for time-sensitive traffic. For instance, in [40] a higher priority class is introduced such that the traffic of that class does not go through the CBSA shaper. This type of traffic is called Scheduled Traffic (ST) and is having its transmissions planned offline.

Ethernet POWERLINK

Beside the solutions based on enhanced Ethernet switches, there are also solutions that are proposed based on overlay protocols to control the traffic submitted to the COTS switches. One of these solutions is Ethernet POWERLINK [10] that uses a master-slave technique. The protocol supports both periodic and aperiodic messages using two isolated phases in cyclic time slots. Each cycle is composed of four phases including Start phase, Isochronous phase, Asynchronous phase and Idle phase. In the Start phase, the master node sends a synchronization message to notify the slave nodes about the starting of a new cycle. The master then polls the periodic messages during the Isochronous phase by sending a polling request to each of the messages. Progressively, the slave nodes reply by a polling response that is broadcasted to all nodes. Within the Asynchronous phase, the master sends a particular message, named Start of Asynchronous (SoA) to a specific node. Then, the node replies back with the asynchronous message. At the end of the cycle, an idle phase is considered to enforce a precise cycle start.

2.4 Schedulability Analysis

As the major requirement of the real-time systems is to provide actions in a timely manner, an accurate prediction of timing behavior is required. In or-

der to guarantee that the actions meet their respective deadlines, schedulability analysis techniques have been developed [41] [42] [43]. This section describes some of the analysis techniques commonly used in distributed real-time systems.

2.4.1 Response Time Analysis (RTA)

One of the schedulability analysis techniques is based on calculating the response time of the tasks/messages and comparing that to their deadlines. Response time of a message is the time interval between the activation time of a message in the source node and the reception time in the destination node. This technique is known as Response Time Analysis (RTA) [18]. The RTA technique is a well-known method in the real-time community and it has been used in many real-time domains including microprocessors, multi-core systems and distributed systems. When using this technique the response time of a task/message is calculated iteratively with a pseudo-polynomial run-time complexity, with the aim to find the worst-case scenario for the task/message. Also, interference of other tasks/messages and their activation frequencies are taken into account in the computation.

In the context of single-core microprocessors and assuming a fixed-priority scheduling, the worst-case scenario for a task occurs when all higher priority tasks are released simultaneously with the task, known as critical instant for the task [44]. It has been shown in [45] that in the above mentioned situation, the response time R_i of a task τ_i is calculated by the sum of the task computation time C_i, the interference from the higher priority tasks denoted by I_i and blocking due to sharing a common resource with τ_i by a lower priority task denoted by B_i. Equation (2.1) shows the response time calculation for τ_i.

$$R_i = C_i + B_i + I_i \tag{2.1}$$

Moreover, the interference from higher priority tasks is calculated in Equation (2.2) considering all activations of higher priority tasks during the task τ_i response time, where $hp(i)$ is the set of higher priority tasks.

$$I_i = \sum_{\forall j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j \tag{2.2}$$

Combining equations (2.1) and (2.2) derives an iterative equation shown in (2.3), where $R_i^0 = C_i$ and the result is derived when $R_i^{n+1} = R_i^n$.

$$R_i^{n+1} = C_i + B_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{R_i^n}{T_j} \right\rceil C_j \tag{2.3}$$

Note that, the response time calculation presented above is valid for fixed-priority preemptive scheduling. In case of Earliest Deadline First (EDF) scheduling, the computation is different and presented in [46].

The same algorithms can be applied in the network domain considering different interference, blocking and critical instants. In the work presented in [47] calculation of worst-case response time of messages in the Controller Area Network (CAN) is presented. In the area of Ethernet AVB networks, the work presented in [48] computes the response time of traffic class A and B. Furthermore, the work in [19] and [12] use response time analysis techniques to compute the worst-case response times of Ethernet messages in a switched Ethernet network.

2.4.2 Network Calculus (NC)

Network Calculus [37] is a framework for analyzing deterministic queuing systems based on *min-plus* algebra. The traffic transmitted to a network is affected by constraints imposed by the network components such as link bandwidth and traffic shapers. The constraints associated to the traffic flow are expressed by an *arrival curve*, whereas the availability of resources such as crossed nodes is described by a *service curve*. The delay bound which represents an upper bound to the worst-case response time and the backlog that represents an upper bound to the maximum queue length are computed using the arrival and service curves. Figure 2.2 shows the service and arrival curves.

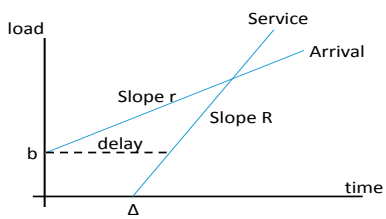

Figure 2.2: The Service and Arrival Curves

In the simplified case the arrival curve can be upper bounded by $\alpha(t) = b + r \times t$ such that b is the maximum burst and r is the rate of traffic. Moreover, the service curve for a node is $\beta(t) = max(0, R \times (t - \Delta))$, where R is the rate of service and Δ is the latency imposed by the node. Then, the delay bound is the maximum distance between the two curves as presented in Figure 2.2. Note that the illustration is a simplified version of the delay calculation, however, the arrival and service curves can be more complex, e.g., step-like curves.

In the context of Ethernet networks, several approaches use Network Calculus to compute the delay of traffic. In the work presented in [49], Network Calculus is used to analyze the delays of the traffic in a single-master and multi-switch network topology using the FTT-SE protocol. Also, the Network Calculus is used in networks of standard Ethernet switches presented in [50]. In the work presented in [51] three methods are used to derive the traffic delays in a multi-hop AFDX network. These three methods include Network Calculus, network simulation and model checking, among which Network Calculus exhibits a higher pessimism.

Focusing on Ethernet AVB, the work presented in [52] utilizes the Network Calculus method to derive traffic delays. Also, the work in [53] presents a worst-case delay verification of in-vehicle Ethernet networks using the same analytical framework to generate upper bounds and checking them against experiments in worst-case scenarios.

A different approach is followed in [54] and [55] that derives end-to-end delay bounds for a single flow in FIFO multiplexed sink-tree networks using a modified Network Calculus framework. Furthermore, the work in [56] utilizes the mentioned method to investigate an admission control in sink-tree networks.

2.4.3 Trajectory Approach

The trajectory approach [38] has been developed to compute the upper bound on delay of the traffic. It is based on analysis of the worst-case scenario experienced by a message on its trajectory from the source node to the destination node. According to this technique, the delay experienced by a message from its release in source node until its reception in destination node, known as end-to-end delay, is the sum of transmission delay and the time the message is spent in each node. Note that, a message can traverse through several intermediate nodes (these nodes can be translated to switches in the context of switched networks). The transmission delay on links is upper bounded by L_{max} that depends on the transmission speed, packet length and physical limitations.

The delay of crossing a node depends on the status of the output queue of the node. Assuming that the queue is FIFO, the arrival message should wait for all messages ahead of that in the queue. In order to compute the waiting time in the crossing nodes, the trajectory approach introduces a *busy period* level $\mathcal{L}$. A busy period [57] is an interval of time between two consecutive *idle times*. An idle time is a time when all previously arrived messages have been processed.

The trajectory approach calculates the latest starting time of message m in its last node considering that the message is generated at time t in its source node. Then, it derives the end-to-end delay.

The trajectory approach has been used in AFDX networks as a tighter analysis compared to the Network Calculus framework [58] [59] [60]. However, in the work presented in [61], the authors showed that for some special cases the improved trajectory approach introduces some optimism, even though these special cases do not exist in any AFDX configuration. The solution to resolve this problem is proposed in [62]. Moreover, the trajectory approach is used to compute the worst-case backlog in the AFDX network [63].

In our research we have used response time analysis to compute the worst-case delay of the traffic, as Network Calculus showed higher pessimism in similar cases (see e.g., [64]). Moreover, applying the trajectory approach on the FTT-SE and HaRTES architectures requires changes in the traffic and network model which is interesting to compare with the response time analysis. This comparison has not been done yet and it remains as future work.

Chapter 3

The FTT-SE and HaRTES Architectures

In this chapter, the basics of the FTT-SE and HaRTES architectures are described. More details of these two architectures are presented in Paper A and Paper B.

3.1 The FTT-SE Architecture

The FTT-SE [65] [11] uses a master-slave technique to coordinate all traffic in the network. This architecture supports periodic, sporadic and non-real-time traffic. The former is classified as *synchronous traffic*, which is time-triggered, and the two latter are categorized as *asynchronous traffic*, which is based on an event-triggered manner. An example of the FTT-SE architecture is depicted in Figure 3.1.

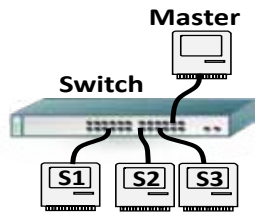

Figure 3.1: A Network Example

A protocol is developed to schedule and transmit the messages in the FTT-SE architecture. According to this protocol, the master node organizes the traffic transmission in fixed-duration time-slots called Elementary Cycle (EC). The data communication in each EC is divided into two specific windows to handle synchronous and asynchronous traffic, which is called the synchronous window and asynchronous window respectively. Figure 3.2 shows the EC and the communication windows in the FTT-SE architecture.

The master node schedules the traffic online according to a user defined scheduling policy, e.g., the Fixed Priority Scheduling (FPS) policy, considering the dedicated window within the EC. If a message fits in the transmission window, the master encodes that into a particular message called Trigger Message (TM) to be broadcasted to the slave nodes at the beginning of the EC (Figure 3.2). Once the nodes receive the TM, they decode it and initiate the transmission of the scheduled messages. The process of decoding and initiating the transmission takes an amount of time that depends on the processing speed of the slave nodes. This amount of time is called turn around time (TRD) that is illustrated in Figure 3.2.

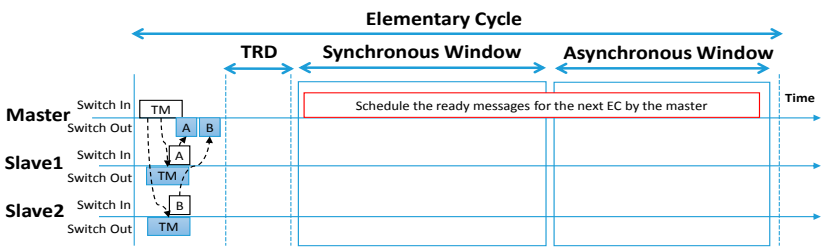

Figure 3.2: The EC in the FTT-SE Protocol

The asynchronous messages make use of a signaling mechanism [66] that allows the master to become aware of them and to consider them in its internal traffic scheduling. The signaling mechanism is based on so-called signaling message (SIG) that is sent by the nodes to the master node, informing it of the status of the nodes' queues (e.g., messages A and B in Figure 3.2). Whenever an asynchronous message becomes active in one node, this node informs the master in the next SIG message that it sends to the master, thus the master schedules the asynchronous message for the upcoming ECs.

In this architecture, the large messages are automatically fragmented into several packets that are scheduled sequentially by the master node.

The main aim of the master node is to schedule the traffic online without causing overrun in the EC, i.e., the scheduled traffic must be received before the end of the EC. This is important in order to enforce the master traffic scheduling policy without causes interference to the management of queues. The scheduler in a master node picks the messages from the ready queue and checks whether they fit in their dedicated window in that EC. This procedure continues until the last message in the ready queue. The unscheduled messages are kept in the ready queue for the next ECs.

To keep track of the utilization of the windows in each link connecting to the switch (assuming common full duplex switches), the master considers two bins per link and per window, one bin representing the uplink (node to switch) and the other bin the downlink (switch to node). Then, the scheduler starts from the higher priority messages and fills up the bins associated to the links in the message path while considering the delays imposed by the switching itself, namely store-and-forward and switch latency delays, and the interference caused by the other traffic.

Let us assume an example that is depicted in Figure 3.1. Also, assume that two synchronous messages, m_1 and m_2, are ready to be transmitted from node S1 to node S2 and from node S3 to node S2, respectively. These two messages share a link to node S2.

Figure 3.3 illustrates the bins corresponding to the uplink of S1 and S3, and the downlink of S2. Assuming m_1 has higher priority than m_2, the scheduler fills up first the uplink S1 bin with m_1 transmission time. It also fills up the downlink S2 bin with the message transmission time plus its switching delay resulting from crossing the switch. Then the scheduler checks m_2 and fills up the uplink of S3 and the downlink of S2, the same way for m_1. If both messages fit in the bin, the scheduler encodes them into the TM for broadcasting to the nodes. In case m_2 does not fit in one of the respective bins, the scheduler keeps that in the ready queue for the next EC scheduling.

3.2 The HaRTES Architecture

The HaRTES switch [67] [12] is a modified Ethernet switch based on a master-slave technique where the master is implemented inside the switch. Figure

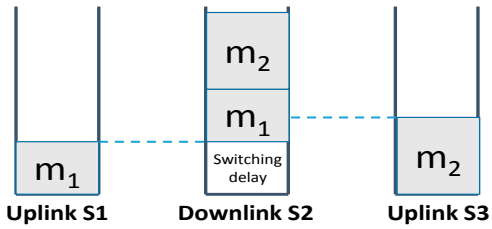

Figure 3.3: The Allocated Bins for the Example

3.4 depicts the abstract functional structure of the switch. The Packet Classification module, implemented for each input port, is analyzing the incoming packets to distinguish the traffic types. Then, the module appends the traffic to the memory queue in the Memory Pool module. The Master module contains the scheduler, admission control, QoS management and a repository of the traffic attributes. These attributes include the deadline, minimum inter-arrival time/period, message length and priority.

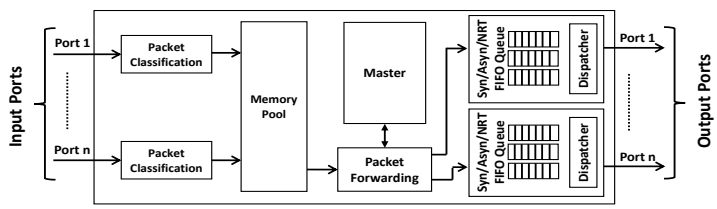

Figure 3.4: HaRTES Switch Structure

For each output port three FIFO queues are implemented to handle synchronous, asynchronous and non-real-time packets, which are identified as Syn, Asyn and NRT respectively in Figure 3.4. Finally, the Dispatcher module in the output port allows the traffic transmission within the associated windows.

The HaRTES switch generates two types of delays, known as store-and-forward and fabric latency. The former corresponds to the time required to receive the message before forwarding it, whereas the latter delay is due to the processing speed of the switch.

Similar to the FTT-SE architecture, the master module is responsible for scheduling the traffic within the EC. Also, the EC is divided between two win-

dows, one for scheduling of synchronous traffic and the other one for asynchronous traffic. Controlling the transmission within these windows is performed by the Dispatcher module.

In each EC, the switch determines new activation of the synchronous messages, it checks whether the message can be transmitted within the coming EC. The scheduled messages are encoded into the TM, to be transmitted to the nodes at the beginning of the EC. The slave nodes receive the TM, decode it and initiate the transmission within the associated window.

Unlike the FTT-SE architecture, the asynchronous messages are transmitted to the switch autonomously without triggering by the TM. The switch buffers the asynchronous messages and schedules them for the next ECs. Note that, the asynchronous messages are not allowed to be sent within the synchronous window. Thus, during the synchronous window, such messages are buffered in the switch. This way the temporal isolation among the synchronous and asynchronous messages are achieved.

3.3 FTT-SE vs HaRTES

Despite the similarities between FTT-SE and HaRTES, there are also subtle but important differences, which have a strong impact in the operation and performance of both protocols. In particular, the master module is inserted in the HaRTES switch to overcome some limitations such as capability of temporal control by the switch. In the FTT-SE architecture, the nodes should be FTT-compliant to respect the timing of the EC. This requires a specific network driver and legacy nodes may not respect the protocol timing. By adding the master inside the switch the traffic confinement is achieved even for the legacy nodes.

Moreover, due to having a master module inside the HaRTES switch, some unauthorized real-time traffic can be blocked to prevent the interference to the other traffic. Also, non FTT-compliant nodes can be connected to the HaRTES switch without jeopardizing the real-time services. This is in fact due to having traffic confinement in the HaRTES switch.

In the HaRTES switch, the asynchronous messages are transmitted autonomously without being triggered by the master module, while in the FTT-SE architecture the asynchronous messages are polled by the master node. This eliminates the signaling mechanism in the HaRTES architecture, hence reduces the scheduling algorithm complexity.

Another difference is regarding the reserved bandwidth in different links.

In the HaRTES architecture it is possible to have different bandwidth in different links, according to the actual load. This property results in more efficient bandwidth usage.

Furthermore, unlike the COTS switches, HaRTES has ability to buffer the traffic due to implementation of a memory pool in the switch. Therefore, it is possible to buffer the traffic for a few ECs before transmitting to another switch or node.

Beside the advantages of the HaRTES architecture, the FTT-SE architecture uses COTS Ethernet switches. Due to the wide availability of standard switches, the FTT-SE architecture provides a low cost solution compared to the HaRTES architecture.

We have used these particular properties when investigating the extended protocols for multi-hop communication.

Chapter 4

Conclusions

4.1 Summary

In this thesis, we extended two particular switched Ethernet-based technologies, the FTT-SE and HaRTES architectures, to support large scale multi-hop networks.

First, we proposed two different architectures to extend the FTT-SE architecture for multiple switches, named the multi-master and the hybrid architectures. We presented response time analyses for the architectures and compared them with respect to bandwidth utilization. We showed that the hybrid architecture is more suitable for large scale networks (Paper A).

Thereafter, we focused on the HaRTES architecture and proposed two solutions to support multi-hop communication. The first solution (i.e., the DGS) is based on a distributed scheduling approach where each switch is responsible to schedule its own traffic (Paper B). The second solution (i.e., the RBS) is also based on a distributed scheduling approach with more freedom in the traffic transmission compared to the previous solution. As a consequence, it can deliver the traffic to the destination faster than the DGS approach in most of the cases (Paper C). We presented a response time analysis for each solution and compared them based on their response time analyses. We showed that the RBS solution performs better in most of the cases compared to the DGS solution. However, the DGS solution has its advantages such as lower implementation complexity.

Furthermore, we presented a clock synchronization solution in order to timely synchronize the master nodes in the FTT-SE architecture and we de-

picted the effects of this solution on the traffic transmission and the bandwidth utilization (Paper D). We showed that this solution performs better, in particular for large scale networks, rather than the synchronization techniques presented in Paper A to C. Although the clock synchronization solution is presented in the context of the FTT-SE architecture, it can be directly applied on the HaRTES architecture as both architectures are based on the FTT paradigm with a similar master-slave scheduling technique. Finally, we developed a simulation tool, named Switched Ethernet Simulator (SEtSim), that we used to evaluate different switched Ethernet-based architectures (Paper E).

4.2 Future Work

In general, the proposed solutions to support multi-hop communication over the FTT-SE and HaRTES architectures are not fully implemented. Therefore, the main ongoing work is to develop the presented solutions on hardware and experimentally validate them.

Furthermore, there are some pessimism in the response time analyses of all the presented solutions. This pessimism is discussed in details in the published papers (Paper A, B and C) and some sources of pessimism are already identified. However, the solution to remove this pessimism has not been investigated yet, which is another ongoing work.

Regarding the clock synchronization, the proposed protocol has not been implemented using real hardware. Therefore, validating the protocol and experimentally showing its effect on the overall performance of the system is remaining as future work.

One of the key characteristics of the FTT-SE and HaRTES architectures is their adaptive reconfiguration. A middleware is proposed that contains a QoS management and an admission control in the single switch case. Also, some work has been done to extend that for the multi-hop FTT-SE architecture [68] [69]. However, the implementation of the dynamic reconfiguration techniques and experimental evaluations are remaining as future work.

Moreover, the performance of the multi-hop FTT-SE architecture compared to the multi-hop HaRTES architecture is not fully investigated. This direction of future work is also necessary to draw the benefits of each architecture in different applications.

Another future direction is applying the studied methods to other switched Ethernet-based protocols. For instance, the proposed multi-hop communication in HaRTES (e.g., the RBS method) can be tuned to be applied on multi-hop

networks based on COTS Ethernet switches. Also, the proposed response time analysis methods can be tuned for use in other technologies such as Ethernet AVB networks.

Finally, the simulation tool (SEtSim) which is developed to evaluate different architectures can be upgraded in many different aspects. For instance, we can accommodate the Ethernet AVB technology along with the clock synchronization used in the protocol. Therefore, it can provide a framework to compare switched Ethernet technologies together to sketch advantages of each in different applications.

Bibliography

[1] F. Gomez-Molinero. Real-time requirement of media control applications. In *19th Euromicro Conference on Real-Time Systems (ECRTS)*, July 2007.

[2] C.-S. Cho, B.-M. Chung, and M.-J. Park. Development of real-time vision-based fabric inspection system. *IEEE Transactions on Industrial Electronics*, 52(4):1073–1079, August 2005.

[3] A. Kumar. Computer-vision-based fabric defect detection: A survey. *IEEE Transactions on Industrial Electronics*, 55(1):348–363, January 2008.

[4] C.-L. Hwang and C.-Y. Shih. A distributed active-vision network-space approach for the navigation of a car-like wheeled robot. *IEEE Transactions on Industrial Electronics*, 56(3):846–855, March 2009.

[5] H.-T. Lim, L. Volker, and D. Herrscher. Challenges in a future IP/Ethernet-based in-car network for real-time applications. In *48th ACM/EDAC/IEEE Design Automation Conference (DAC)*, 2011.

[6] H.-T. Lim, K. Weckemann, and D. Herrscher. performance study of an in-car switched ethernet network without prioritization. In *Proceedings of the Third international conference on Communication technologies for vehicles*. Springer-Verlag, 2011.

[7] H. Kopetz, A. Ademaj, P. Grillinger, and K. Steinhammer. The time-triggered Ethernet (TTE) design. In *8th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*, May 2005.

[8] W. Steiner, G. Bauer, B. Hall, M. Paulitsch, and S. Varadarajan. TTEthernet dataflow concept. In *8th IEEE International Symposium on Network Computing and Applications*, 2009.

[9] Z. Hanzalek, P. Burget, and P. Sucha. Profinet IO IRT message scheduling. In *21st Euromicro Conference on Real-Time Systems (ECRTS)*, 2009.

[10] Ethernet POWERLINK Standardisation Group. *EPSG Draft Standard 301 Ethernet POWERLINK Communication Profile Specification Version 1.2.0*, 2013.

[11] R. Marau, L. Almeida, and P. Pedreiras. Enhancing real-time communication over COTS Ethernet switches. In *6th IEEE International Workshop on Factory Communication Systems (WFCS)*, June 2006.

[12] R. Santos, M. Behnam, T. Nolte, P. Pedreiras, and L. Almeida. Multi-level hierarchical scheduling in ethernet switches. In *Proceedings of the International Conference on Embedded Software (EMSOFT)*, October 2011.

[13] SERV-CPS project, available at http://serv-cps.av.it.pt/, access date October 2014.

[14] R. Marau, L. Almeida, M. Sousa, and P. Pedreiras. A middleware to support dynamic reconfiguration of real-time networks. In *IEEE Conference on Emerging Technologies and Factory Automation (ETFA)*, September 2010.

[15] R. Marau, L. Almeida, P. Pedreiras, K. Lakshmanan, and R. Rajkumar. Utilization-based schedulability analysis for switched Ethernet aiming dynamic QoS management. In *15th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, September 2010.

[16] P. Pedreiras and L. Almeida. The flexible time-triggered (FTT) paradigm: an approach to QoS management in distributed real-time systems. In *The International Parallel and Distributed Processing Symposium (IPDPS)*, April 2003.

[17] F. Yekeh, M. Pordel, L. Almeida, M. Behnam, and P. Portugal. Exploring alternatives to scale FTT-SE to large networks. In *6th IEEE International Symposium on Industrial Embedded Systems (SIES)*, June 2011.

[18] M. Joseph and P. Pandya. Finding response times in a real-time system. *The Computer Journal*, 29(5):390–395, March 1986.

[19] R. Marau, M. Behnam, Z. Iqbal, P. Silva, L. Almeida, and P. Portugal. Controlling multi-switch networks for prompt reconfiguration. In *Proceeding of 9th International Workshop on Factory Communication Systems (WFCS)*, May 2012.

[20] K.-G. Larsen, P. Pettersson, and W. Yi. Uppaal in a nutshell. *International Journal on Software Tools for Technology Transfer*, 1997.

[21] R. Snieder and K. Larner. *The Art of Being a Scientist: A Guide for Graduate Students and their Mentors*. Cambridge University Press, 2009.

[22] M. Barr and A. Massa. Programming embedded systems, 2nd edition. In *O'Reilly Media*, October 2006.

[23] R. N. Charette. This car runs on code. In *Spectrum, IEEE, 46(2)*, February 2009.

[24] M. Broy, IH. Kruger, A. Pretschner, and C. Salzmann. Engineering automotive software. *Proceedings of the IEEE*, 95(2):356–373, February 2007.

[25] H. Kopetz. *Real-Time Systems: Design Principles for Distributed Embedded Applications*. Kluwer Academic Publishers, 1st edition, 1997.

[26] *The FlexRay Communication System Specification, version 2.1*. 2005.

[27] L. Almeida, P. Pedreiras, and J.AG. Fonseca. The FTT-CAN protocol: why and how. *IEEE Transactions on Industrial Electronics*, 49(6):1189–1201, December 2002.

[28] P. Pedreiras, P. Gai, L. Almeida, and G. Buttazzo. FTT-Ethernet: a flexible real-time communication protocol that supports dynamic QoS management on Ethernet-based systems. *IEEE Transactions on Industrial Informatics*, 1(3):162–172, August 2005.

[29] L. Sha and S.S. Sathaye. A systematic approach to designing distributed real-time systems. *IEEE Computer*, 26(9):68–78, September 1993.

[30] G. Buttazzo. *Hard Real-Time Computing Systems, third edition*. Springer, 2011.

[31] H. Hoang and M. Jonsson. Switched real-time Ethernet in industrial applications - deadline partitioning. In *9th Asia-Pacific Conference on Communications (APCC)*, September 2003.

[32] S. Varadarajan and T. Chiueh. EtheReal: a host-transparent real-time fast ethernet switch. In *6th International Conference on Network Protocols*, October 1998.

[33] H. Kopetz, A. Ademaj, P. Grillinger, and K. Steinhammer. The time-triggered Ethernet (TTE) design. In *8th IEEE International Symposium on Object-Oriented Real-Time Distributed Computing*, May 2005.

[34] *Profibus International, Application layer protocol for decentralized periphery and distributed automation, specification for PROFINET, IEC 61158-6-IO/FDIS*. October 2007.

[35] *Profibus International, Application layer services for decentralized periphery and distributed automation, specification for PROFINET, IEC 61158-5-IO/FDIS*. October 2007.

[36] I. Land and J. Elliott. *Architecting ARNIC 664 (AFDX) Solutions*. 2011.

[37] J. Leboudec and P. Thiran. *Network Calculus*. Berlin, Germany: Sprige-Verlag, 2001.

[38] S. Martin and P. Minet. Schedulability analysis of flows scheduled with FIFO: application to the expedited forwarding class. In *20th International Parallel and Distributed Processing Symposium*, April 2006.

[39] Audio/video bridging task group of ieee 802.1, available at http://www.ieee802.org/1/pages/avbridges.html.

[40] G. Alderisi, G. Patti, and L.L. Bello. Introducing support for scheduled traffic over IEEE audio video bridging networks. In *IEEE 18th Conference on Emerging Technologies Factory Automation (ETFA)*, September 2013.

[41] L. Sha, T. Abdelzaher, K.-E. rzn, A. Cervin, T. Baker, A. Burns, G. Buttazzo, M. Caccamo, J. Lehoczky, and A. K. Mok. Real time scheduling theory: A historical perspective. *Real-Time Systems*, 28(2):101–155, February 2004.

[42] N. Audsley, A. Burns, M. Richardson, K. Tindell, and A.J. Wellings. Applying new scheduling theory to static priority pre-emptive scheduling. *Software Engineering Journal*, 8(5):284–292, September 1993.

[43] N. C. Audsley, A. Burns, R. I. Davis, K. W. Tindell, and A. J. Wellings. Fixed priority pre-emptive scheduling: An historical perspective. *Real-Time Systems*, 8(2):173–198, March/May 1995.

[44] C.L. Liu and J.W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of ACM*, 20(1):46–61, 1973.

[45] N.C. Audsley, A. Burns, M. F. Richardson, and A. J. Wellings. Hard real-time scheduling: The deadline-monotonic approach. In *IEEE Workshop on Real-Time Operating Systems and Software*, pages 133–137, 1991.

[46] m. Gonzalez Harbour and J.C. Palencia. Response time analysis for tasks scheduled under EDF within fixed priorities. In *24th IEEE Real-Time Systems Symposium (RTSS)*, December 2003.

[47] R. I. Davis and A. Burns. Controller area network (CAN) schedulability analysis: Refuted, revisited and revised. *Real-Time Systems*, 35:239–272, 2007.

[48] U. D. Bordoloi, A. Aminifar, P. Eles, and Z. Peng. Schedulability analysis of Ethernet AVB switches. In *The 20th IEEE International Conference on embedded and Real-Time Computing Systems and Applications (RTCSA)*, August 2014.

[49] A. Mifdaoui, F. Frances, and C. Fraboul. Performance analysis of a master/slave switched Ethernet for military embedded applications. *IEEE Transactions on Industrial Informatics*, 6(4), November 2010.

[50] M. Zhang, J. Shi, T. Zhang, and Y. Hu. Hard real-time communication over multi-hop switched Ethernet. In *The 2008 IEEE International Conference on Networking, Architecture, and Storage (NAS)*, June 2008.

[51] H. Charara, J.-L. Scharbarg, J. Ermont, and C. Fraboul. Methods for bounding end-to-end delays on an AFDX network. In *18th Euromicro Conference on Real-Time Systems (ECTRS)*, July 2006.

[52] R. Queck. Analysis of Ethernet AVB for automotive networks using network calculus. In *IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, July 2012.

[53] M. Manderscheid and F. Langer. Network calculus for the validation of automotive Ethernet in-vehicle network configurations. In *International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, October 2011.

[54] L. Lenzini, L. Martorini, E. Mingozzi, and G. Stea. Tight end-to-end per-flow delay bounds in FIFO multiplexing sink-tree networks. *Elsevier Performance Evaluation*, 63, October 2006.

[55] J.B. Schmitt, F.A. Zdarsky, and M. Fidler. Delay bounds under arbitrary multiplexing: When network calculus leaves you in the lurch... In *The 27th IEEE Conference on Computer Communications*, April 2008.

[56] L. Lenzini, L. Martorini, E. Mingozzi, and G. Stea. A novel approach to scalable CAC for real-time traffic in sink-tree networks with aggregate scheduling. In *the 1st international conference on Performance evaluation methodolgies and tools*. ACM, October 2006.

[57] J.P. Lehoczky. Fixed priority scheduling of periodic task sets with arbitrary deadlines. In *11th Real-Time Systems Symposium*, December 1990.

[58] H. Bauer, J.-L. Scharbarg, and C. Fraboul. Applying and optimizing trajectory approach for performance evaluation of AFDX avionics network. In *The 14th IEEE Conference on Emerging Technologies Factory Automation (ETFA)*, September 2009.

[59] H. Bauer, J.-L. Scharbarg, and C. Fraboul. Improving the worst-case delay analysis of an AFDX network using an optimized trajectory approach. *IEEE Transaction on Industrial Informatics*, November 2010.

[60] H. Bauer, J.-L. Scharbarg, and C. Fraboul. Applying trajectory approach with static priority queuing for improving the use of available AFDX resources. *Real-Time Systems Journal*, 48(1):101–133, 2012.

[61] G. Kemayo, F. Ridouard, H. Bauer, and P. Richard. Optimistic problems in the trajectory approach in FIFO context. In *18th IEEE Conference on Emerging Technologies Factory Automation (ETFA)*, September 2013.

[62] X. Li, O. Cros, and L. George. The trajectory approach for AFDX FIFO networks revisited and corrected. In *The 20th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA)*, August 2014.

[63] H. Bauer, J.-L. Scharbarg, and C. Fraboul. Worst-case backlog evaluation of avionics switched ethernet networks with the trajectory approach. In *24th Euromicro Conference on Real-Time Systems (ECRTS)*, July 2012.

[64] M. Ashjaei, M. Behnam, L. Almeida, and T. Nolte. Performance analysis of master-slave multi-hop switched ethernet networks. In *8th IEEE International Symposium on Industrial Embedded Systems (SIES)*, June 2013.

[65] R. Marau. *Real-time communications over switched Ethernet supporting dynamic QoS management*. PhD Thesis, University of Aveiro, Aveiro, Portugal, 2009.

[66] R. Marau, P. Pedreiras, and L. Almeida. Asynchronous traffic signaling over master-slave switched ethernet protocols. In *6th International Workshop on Real Time Networks (RTN)*, July 2007.

[67] R. Santos. *Enhanced Ethernet Switching Technology for Adaptive Hard Real-Time Applications*. PhD Thesis, University of Aveiro, Aveiro, Portugal, 2010.

[68] M. Ashjaei, P. Pedreiras, M. Behnam, L. Almeida, and T. Nolte. Dynamic reconfiguration in multi-hop switched Ethernet networks. In *6th Workshop on Adaptive and Reconfigurable Embedded Systems*, April 2014.

[69] M. Ashjaei, P. Pedreiras, M. Behnam, L. Almeida, and T. Nolte. Evaluation of dynamic reconfiguration architecture in multi-hop switched Ethernet networks. In *The 19th IEEE International Conference on Emerging Technologies and Factory Automation, Work-in-Progress (WiP) session*, September 2014.