Nancy-Université INPL

Institut National Polytechnique de Lorraine

ÉCOLE DOCTORALE IAEM

Département de formation doctorale en informatique

T H È S E

présentée et soutenue publiquement le

pour l'obtension du

Doctorat de l'Institut National Polytechnique de Lorraine (spécialité informatique)

par

Dawood A. KHAN

Schedulability Analyses for the Design of Reliable and Cost-effective Automotive Embedded Systems

Thèse dirigée par Françoise SIMONOT-LION et Nicolas NAVET

préparée au INRIA Grand-Est, Projet TRIO

Jury :

Rapporteurs :			
	Emmanuel GROLLEAU	-	Professeur à LISI/ENSMA
	Jean-Luc SCHARBARG	-	MC à Université de Toulouse
Examinateur :	Yvon TRINQUET	-	Professeur à Université de Nantes
	Sylvain CONTASSOT-VIVIER	-	Professeur à LORIA/UHP

Loria Laboratoire lorrain de recherche en informatique et ses applications

Laboratoire Lorrain de Recherche en Informatique et ses applications – UMR7503

Abstract

Automotive embedded systems are distributed architectures of computer-based applications. These automotive embedded systems have brought many benefits such as replacement of old mechanical system with wired ones and new applications ones like adaptive suspensions. These replacements or enhancements could be of critical nature and therefore providing guarantees that these embedded systems will to perform, even in harsh environments, is of utmost importance. Besides, these computer-based applications demand timeliness, imposed by a physical process. For example, braking subsystem is usually spread over many embedded nodes which are communicating with each other over a shared resource and has time constraints that need to be met. Therefore, it is important that time constraints are met individually and collectively in the composition of these embedded nodes. That is the time between the brake application at brake pedal to the brake actuation at the wheels of an automobile, the time duration should be less than the deadline. Moreover, such a proliferation has also come with an increasing heterogeneity and complexity of the embedded architecture.

Therefore, there is a need to ensure that these automotive embedded systems meet temporal constraints, and provide safety guarantees during normal operation or critical situations. This thesis aims at developing the schedulability analyses for automotive systems and embedded networks, with the aim to facilitate cost-effective and reliable design and analysis of automotive embedded systems. The analyses are applied/developed in the automotive domain, to reduce the risk of deadline failure due to hardware limitations, implementation overheads and interference due to probabilistic traffic.

Keywords: controller area network, CAN, real-time communication, real-time analysis, scheduling, probabilistic analysis, component based system

Collaborations

Following is a list of people with whom I have done research, co-authored papers, or generally worked, on research problems:

- Riender J. Bril, Technical University Eidhoven: On the initial part of Chapter 3 dealing with integration of copy-time into the CAN schedulability analysis.
- Robert I. Davis, University of York: On the later part of Chapter 3 dealing with integration of non-abortable transmission into the CAN schedulability analysis.
- Luca Santinelli, TRIO, INRIA Grand Est: On the analysis framework developed in Chapter 4.

Contents

Chapter 1

Introduction

Contents

1.1 Introduction

Automotive embedded systems are distributed architectures of computer-based applications with physical processes (mechanical, hydraulic) that they have to control. The growth in proliferation of computers (ECU, Electronic Control Unit) has an impact on the safety. The increased use of ECUs in modern automotive systems has brought many benefits such as the merging of chassis control systems for active safety with passive-safety systems. Most of the automotive applications are safety critical and therefore providing guarantees for these applications is an important requirement. Moreover, such a proliferation has come with an increasing heterogeneity and complexity of the embedded architecture. Therefore, there is a growing need to ensure that automotive embedded systems have reliability, availability and safety guarantees during normal operation or critical situations (e.g. airbags during collision), taking into account harsh environment (heat, humidity, vibration, electro-static discharge ESD and electro-magnetic interference EMI).

To provide guarantee on safety property, model based approaches, and analytical methods during the design activity are required. These approaches should be able to model these systems, which are heterogeneous by nature: discrete and continuous systems, deterministic and probabilistic variables. In particular, to validate timing properties imposed by the time constraints of the physical systems and their

control laws is of utmost importance. The distribution of such systems increases the validation of these safety properties.

Electronic systems in the automobiles are required to respond in a predictable manner, i.e. timely manner. The predictability of these systems is ensured, among others, by timing verification on system models, which checks if performance requirements like deadlines, jitters, throughput etc. are being met.

The timing constraints verification analyses has to be carried out as soon as possible in the development life-cycle. Moreover, such analyses may be mandatory for certification issues.

However, developing timing verification models can be complex to build. We have to find a trade-off between accuracy/complexity/computing time. First, it is difficult to have a detailed model at the earliest step and therefore rough assumptions have to be done on the hardware performances for example. However, such trade-offs should not over-simplify the models thus making the analyses unsafe for use. Analytical timing models, which tend to overlook/oversimplify the system model, may lead to optimistic results that may not fit to the concrete system.

1.1.1 Timing budget

The automotive Original Equipment Manufacturers (OEMs) decomposes the overall end-to-end latency to the timing budget of individual the ECUs, the communication channels, and negotiate these timing budgets with the suppliers. The OEMs need to assign these timing budgets to the suppliers. Therefore, the OEMs must properly decide the time budgets for each ECU and communicate the specification at the initial stage of the automotive development. The OEMs may revise the initial timing estimates of the individual "timing budget" of vehicular functions, to achieve optimal performance or cost of the entire vehicle as the suppliers refine the solution (OEMS may ask suppliers to adjust or improve the time budget). Therefore, OEMs should be able to do better estimates for allocating timing budgets at the initial stages of the projects. The OEMs in practice, therefore, may carry-over from the existing (proven in use) systems with domain-specific rules to estimate the timing budgets, like:

1. The load on an automotive CAN network must not be higher than 30 percent.
2. A frame pending for transmission for more than $30ms$ is canceled out.

However, such an approach has potential problems like being sub-optimal and being unsafe design, with problems that can be hard to reproduce and are costly to repair later in the development cycle. However, we can use the timing information from previous design (of an automotive system) to infer the timing properties of a system in the early stage of design, when very little timing information is available and thus help in better dimensioning of a system. We propose one such model in this thesis, which uses the probabilistic model of aperiodic traffic from previous development run of a vehicle to adjust the aperiodic traffic on a current development run of a vehicle.

1.1.2 Simulations

Simulation is a tool for checking the validity of a system. However, even if the design passes all the tests successfully, it is not necessary that the safety properties will be met. In order to verify worst-case (for safety critical systems) we must perform exhaustive simulations of the design. The simulations utilizes a logical model of system (physical) to imitate state changes in response to random or deterministic events at simulated points in time. The system state changes based on the given system description. For example, in a network to measure the end-to-end response time of messages across the network. In practice software simulations are used in the early stages of development cycle. The simulations are also used to validate analytic models : latencies, buffer occupation, etc. telling us about how long we stay in the worst-case situation. Moreover, the simulations are also performed in conjunction with the ECUs as they become available, HiL (Hardware in the Loop)[1], to validate the system.

However, simulations only cannot be used to do timing verification for the systems with safety and criticality requirements. The reason being the difficulty to ascertain the worst-case from the simulation traces, as they do not provide any bound on the performance results.

1.1.3 Analytical models

The analytical models of automotive systems have been developed and are used to perform timing verifications. These models combine the communication constraints and message specifications (e.g., activations) to do timing verification. The analytical models of the automotive system often consider the periodic and sporadic tasks activations only. For example, analytical models developed for CAN are used to perform timing verification of the messages on CAN bus based on periodic or sporadic activations.

The analytical models have to guarantee that the timing requirements of all tasks are met, i.e. the communications delay between a sending task queuing a message, and a receiving task being able to access that message, must be bounded. This total delay is termed the end-to-end communications delay. The end-to-end communication delay is then used to conclude about the feasibility of the system. Therefore, it is of paramount importance, particularly for safety critical systems, that the upper bound returned by these analyses is a true upper bound.

However, some the analytical models have been prove to be optimistic and thus wrong (especially unpublished complex ones), [Davis 2007], and ignore the impact of hardware limitations and error-proneness of embedded software. Some of the models do the overestimation, which is pessimistic for soft real-time automotive applications.

Moreover, the timing verification models fall short in modeling accurately everything, for example, taking in the account the queuing policy used in device driver,

[1] We do not consider other simulation methods like HiL in this thesis.

copy-time of messages from device driver to communication hardware, limited transmit buffers in a hardware etc. and unfortunately the standards do not say everything about this, e.g., AUTOSAR CAN driver specification.

Moreover, these analytical models do not characterized the network traffic very well e.g. aperiodic traffic. These analyses models usually rely on periodic or sporadic traffic models for pessimistic analysis, based on critical-instance of the tasks/messages in order to find the worst-case timing properties and test the schedulability requirements of the tasks/messages. Even if it is appropriate in some specific application areas, this approach does not allow to address many of the applications in an heterogeneous systems like automobiles; because, when the arrival times are aperiodic with high variance, it may lead to a significant over-provisioning of resources at the design time. Thus for real-time systems (RTS) in which the task/messages set exhibit substantial variability in arrivals (aperiodic), it is practical to develop an approach taking into account the stochastic nature of arrivals of tasks/messages. Such approaches can lead to a drastic reduction in the amount of resource provisioning. Thus leading a system, conceived to be analyzable in temporal domain, to be a potentially unsafe design, which is unacceptable particularly for safety critical automotive systems.

1.2 State of the art

Timing enables an early analysis of whether a system can meet the desired timing requirements, and avoid over- or under- dimensioning of systems and also save from unnecessary iterations in the development process. The result is a shortened development cycle with increased predictability/timeliness, which is of greater interest in safety-critical systems.

Today, during the automotive development process the designers firstly focus on the functional behavior of the system and, therefore, the temporal properties of the systems may be verified late in the process. Besides, when the temporal properties are verified, it is usually through testing and measurements and if a timing error is detected it is late in the process. Therefore, resulting in the costly design re-iterations. Thus, we need the analytical models which we can use from the early stages of the design (not just testing and measurements at the end) to verify timing properties. These analytical models should be detailed enough (for both hardware and software) to check the temporal properties, particularly for safety-critical systems. There are various methods for temporal analyses, which can be broadly grouped into four categories based on the modeling framework they use, and are explained below.

1.2.1 Simulation

The simulations utilizes a logical model of system (physical) to imitate state changes in response to random or deterministic events at simulated points in time. The system state changes based on the given system description. In RTS the Discrete

Event simulation is used to analyze the performance of the system, for example, in a network to measure the end-to-end response time of messages across the network. The transfer time is determined for different bus loads, priorities of the messages and arrangement of the devices. Simulations are often used when an analytical approach is not possible or is complex and expensive. There are various simulation frameworks available for real-time systems and some of which have been described hereafter.

Modeling and Analysis Suite for Real-Time Applications (MAST),see [Gonzalez Harbour 2001] is mixed system providing worst-case schedulability analysis for hard timing requirements, and discrete-event simulation for soft timing requirements. In MAST a system representation is analyzable through a set of tools that have been developed within the MAST suite. These tools describes a model for representing the temporal and logical elements of real-time applications. MAST allows a very rich description of the system, including the effects of event or message-based synchronization, multiprocessor and distributed architectures as well as shared resource synchronization. MAST currently includes only fixed priority scheduling, but, it is conceived as an open model and is easily extensible to accommodate scheduling algorithms.

Ptolemy, see [Buck 2002], is another framework which can provide simulation and prototyping of heterogeneous systems. The models in Ptolemy are described using object-oriented software technology (C++). Ptolemy has been applied to networking and transport, call-processing and signaling software, embedded micro-controllers, signal processing (including implementation in real-time), scheduling of parallel digital signal processors, board-level hardware timing simulation, and combinations of these.

True-Time is toolbox for MATLAB, see [Henriksson 2003], for simulating networked and embedded real-time control systems. One of its main features involves the possibility of co-simulation of the interaction between the real-world continuous dynamics and the computer architecture in the form of task execution and network communication. It supports various communication protocols for both wireless and wired networks.

DRTSS, see [Storch 1996], is another framework which allows its users to easily construct discrete-event simulators of complex, heterogeneous distributed real-time systems. The framework allows simulation of initial high-level system designs to gain insight into the timing feasibility of the system. Which at later stages of design process can be expanded into a detailed hierarchical designs for detailed analysis.

Cheddar, see [Singhoff 2004], is an Ada framework which provides tools to check temporal characteristic of real time applications. The framework is based on the real time scheduling theory. Cheddar model defines an application as a set of processors, tasks, buffers, shared resources and messages. It has a flexible simulation engine which allows the designer to describe and run simulations of specific systems. The cheddar framework is open and extension can be easily designed for tools and simulators.

RTaW-Sim, see [rts], for CAN network is a fine-grained discrete event simulator

providing performance analysis, buffer usage, thereby helping make correct implementation choice e.g. queueing policy. It has features to perform fault-injection in terms of frame transmission errors, ECU reboots, clocks drifting.

Besides these frameworks, simulations in RTS have been used to evaluate the robustness of a system for example, see [Nilsson 2009], where Nilsson et al. created and simulated attacks in the automotive communications protocol FlexRay and showed that such attacks can easily be created. These attacks can result in safety in-vehicle network and lead to serious injury for the driver.

However, it is difficult to ascertain the worst-case from the simulation traces as they do not provide any bound on the performance results. Thus simulations do not qualify for checking temporal properties of hard real-time systems.

1.2.2 Deterministic analyses

The idea of holistic scheduling is to extend well-known results of the classical scheduling theory to distributed systems. These analyses combines the schedulability analyses of processor and communication bus to compute the end-to-end response time in a distributed real-time system. Tindell and Clark in [Tindell 1994a] used this approach to analyze distributed hard real-time system where tasks with arbitrary deadlines communicated by message passing and shared data objects and the nodes communicated via TDMA bus. The developed analysis provided bounds on the communication delays and overheads at the destination processor.

In [Yen 1995, Yen 1998] presented holistic analysis approach for distributed systems where in the described methodology to co-synthesize communication to avoid bottleneck in many embedded systems. They used a bus model for communication with arbitrary topologies in a point-to-point manner. Since, communication links add both chip and board costs, and designers frequently underestimate peak load.

In [Pop 2002] presented a holistic analysis for emerging distributed automotive applications specifically dealing with the issues related to mixed, event-triggered and time-triggered task sets, which communicated over bus protocols consisting of both static and dynamic phases.

However, the problem with holistic scheduling is that it is tailored towards a "particular combination" of input event model, resource sharing policy and communication arbitration. Therefore, for the large heterogeneous systems it results in the large and heterogeneous collection of analyses methods, which makes holistic scheduling analysis difficult to use in practice.

1.2.3 Compositional performance analysis

In contrast to holistic methods that extend classical scheduling analyses, the compositional analyses techniques are modular in nature (components). The components of a system are analyzed with classical algorithms and the local results are propagated in the system through appropriate component interfaces relying on event stream models for propagation between components. That is for each cycle of sys-

tem level compositional analysis, local analysis on each component is performed. The output event models resulting from the local analysis of components are then propagated through the component interface to the connected components. The receiving component uses the output event model from the previous component as its input model.

Thiele et al. in [Thiele 2000] presented Modular Performance Analysis (MPA) as one such analysis method of RTS. The method uses Real-Time Calculus, which is an extension of Network Calculus [Le Boudec 2001], to analyze the flow of event streams through processing and communication elements of the system. The important feature of MPA is that it is not limited to only certain input event models and the component interfaces, see [Henzinger 2006], but can also specify the component compatibility and relationships depending on assumptions about input event model and allocated resource capacities.

SymTA/S (Symbolic Timing Analysis for Systems) is another compositional analysis approach similar to MPA, see [Henia 2005]. The SymTA/S is based on the technique to couple local scheduling analysis algorithms using event streams. Where the event streams describe the possible task activations. For the compositional analysis, the input and output event streams are described by standard event models, for example, a periodic with jitter event model having two parameters can be described as (P, J). SymTA/S compositional approach also has an ability, like greedy shapers in MPA, to adapt the possible timing of events in an event stream.

1.2.4 Probabilistic performance analysis

The worst-case evaluation may not be sufficient or needed as there are not many strict hard real-time systems. Therefore, for these system probabilistic performance analyses are performed. The motivation being that not many applications are time-critical, but nonetheless they are sensitive to latencies. For example, for control applications the quality of the controls depends also on the average response time, besides the deadline, which needs to be minimized. Moreover, the activation of tasks and messages can be aperiodic (probabilistic) in certain system. Importantly, not all of the design parameters may be available at the initial phase of automotive system design and a designer can start with a probabilistic model of a system which can provide an important direction for future phase of the project. Moreover, for many safety critical system the constraints on criticality are represented in terms of the probability thresholds (e.g. mean-time to failure probability).

Stochastic Network Calculus (SNC), see [Jiang 2008], is one such method which focuses on performance guarantees. It is similar to network calculus, a theory dealing with queuing systems found in computer networks, but works with stochastic arrival curves and provides probabilistic guarantees of timing and backlog information. Besides SNC many automotive systems have been analyzed using probabilistic approach, because of problem being explicitly probabilistic in nature. For example, in [Navet 2000], Navet et al. introduce the concept of worst case deadline failure probability (WCDFP), the probability that too many errors occur such that a mes-

sage can not meet its deadline. Nolte et al. in [Nolte 2001] extend the worst-case response time analysis for message with random message transmission times due to bit stuffing. Which depends on the probability distribution of a given number of stuffed bits due to the mechanism in CAN protocol, such that a frame containing a sequence of five consecutive identical bits are "bit-stuffed" to change polarities. Gardner et al. in [Gardner 1999] analyzed a stochastic fixed priority RTS such that an occasional missed deadline is acceptable, but at decreased performance. They presented an analysis technique in which they bound (lower) the percentage of deadlines that a periodic task meets and compared that with the lower bound with simulation results. Diaz et al. in [Díaz 2002] provided a stochastic analysis method for general periodic real-time systems, accurately computing the response time distribution of each task in the system. Which made it possible to determine the deadline miss probability of individual tasks, even for systems with maximum utilization factor greater than one. Bernat et al. in [Bernat 2002] devised an approach for computing probabilistic bound on execution time by combining the measurement and analytical approaches into a model. There method combined, probabilistically, worst-case effects seen to formulate the execution time model of the worst case path of the program.

1.3 Research questions and Contributions

This thesis address the timing verification issues for the automotive systems and provides the analytical models and implementation guidelines to address these problems in a safety critical automotive environment. We investigate and provide tighter worst-case bound in a mixed communication paradigm based on aperiodic (probabilistic) and periodic messages, thus helping in better dimensioning of the systems at the development time. We also investigate the implication of diverse communication controllers (when message abortion is not possible) on response time of the messages that are assumed to be en-queued by the middle-ware-level task before being exchanged on a CAN network and provide a tighter bound on response time of the messages. We also integrate implementation over-heads, such as copy-time, into the schedulability analysis of CAN network. We also develop a probabilistic system-level analysis for component based RTS in a mixed communication paradigm i.e. having both probabilistic and deterministic arrivals. Most of the analyses developed in thesis integrate the concept of functional safety based on Safety Integrity Levels into response time analyses, in order to guarantee the required safety levels. Each chapter provides a case-study which is evaluated using the developed analysis to provide an understanding about improvements and innovations our analyses have brought about. Specifically, this thesis tries to answers the following research question:

- Q1 How to perform mixed (probabilistic and deterministic) timing analysis of an automotive communication network in order to dimension the system properly?

  - Q1a How to model the aperiodic data probabilistically?
  - Q1b How to integrate the model of aperiodic data in the schedulability analysis?
  - Q1c How to ensure that the analysis guarantees the required level of safety?

  Answer: We provide a probabilistic approach to model the aperiodic traffic and integration of it into response time analysis along with the deterministic part, modeled by periodic activations. The approach allows the system designer to choose the safety level of the analysis based on the system's dependability requirements. Compared to existing deterministic approaches the approach leads to more realistic WCRT evaluation and thus to a better dimensioning of the hardware platform.

- Q2 How can different hardware and software implementations affect the temporal behavior in an automotive network?

  - Q2a How to integrate the implementation over-heads in the schedulability analysis?
  - Q2b How to integrate affect of limited transmission buffers in the schedulability analysis?
  - Q2c What are the guidelines for device driver implementations?

  Answer: We provide analysis of the real-time properties of message in a CAN network having hardware constraints and implementation over-heads (copy-time of messages). Which, if not considered, may result in a deadline violation incurred due additional latencies. We explain the cause of this additional latency and extend the existing CAN schedulability analysis to integrate it. We also provide some guidelines that can be useful for the implementation of CAN device drivers.

- Q3 How can we perform a mixed (deterministic and probabilistic) component based performance analysis, for system dimensioning and component reuse, of an automotive system?

  - Q3a How to model the probabilistic component and its interface?
  - Q3b How to compose the mixed (deterministic and probabilistic) components together in a system?
  - Q3c How to do the performance analysis of this mixed component system?
  - Q3d How to ensure that the analysis guarantees the required level of safety?

  Answer: We provide an analysis of complex real-time systems involving component-based design and abstraction models. We developed an abstraction which provides both deterministic and probabilistic models for component interfaces based on curves and probability thresholds associated with

those curves, resulting in an analysis for real-time systems which has both deterministic and probabilistic components, based on an extension of real-time calculus to probabilistic domain. The analysis can offer either hard or soft real-time guarantees according to the requirements and the specifications of the system. We also show the flexibility of the analysis to cope with the required safety criticality level of a system.

1.4 Thesis outline

- Chapter 2: Periodic and Aperiodic (mixed) analysis of CAN based on integrating safety requirements.
- Chapter 3: CAN controller hardware and software limitations and modeling the analysis to include those limitations for tighter bounds on response time.
- Chapter 4: System level response time analysis for component based analysis, in a mixed (probabilistic and deterministic) analysis for system level performance with guarantees for safety and real-time constraints.
- Chapter 5: Gives the perspective of this thesis.

Chapter 2

Probabilistic CAN Schedulability Analysis

Contents

In this chapter a probabilistic approach to model the aperiodic traffic and integration of it into response time analysis is discussed. The approach allows the system designer to choose the safety level of the analysis based on the system's dependability requirements. Compared to existing deterministic approaches the approach leads to more realistic WCRT evaluation and thus to a better dimensioning of the hardware platform.

2.1 Introduction

In the field of real-time systems, methods to assess the real-time performances of periodic activities (tasks, messages) have been extensively studied. Response times, worst-case or average, and jitters can be evaluated by simulation or analysis for a wide range of scheduling policies provided that the activation patterns of the tasks and messages are well identified. The problem is more intricate for aperiodic activities since, in many practical cases, it is difficult to have a precise knowledge of their

activation pattern and because deterministic WCRT analysis have not been conceived to handle aperiodic activities. For example, the arrival pattern of aperiodic frames in the body network of a vehicle is hard to predict, as it is dependent on the user interactions. However aperiodic frames of higher priority exchanged among the Electronic Control Units (ECUs) in the body network of a vehicle can delay periodic traffic. Indeed, most often the Controller Area Network (CAN) priority bus is used and the aperiodic frames do not necessarily get the lowest priority levels [1]assigned to them.

2.1.1 Problem definition

In this chapter, we address the problem of evaluating response times when both periodic and aperiodic activities are taken into account. Activities are termed frames in rest of the chapter, because the approach will be developed and illustrated on the CAN bus, but our approach equally holds for tasks. The increase in the WCRT of the periodic frames which may be caused by the higher priority aperiodic frames could be critical for hard real-time systems as it could lead to the violation of the deadlines. Besides, large response times of aperiodic frames may jeopardize the execution of a function or may even raise safety concerns in some cases (e.g. headlights flashes in a vehicle). In addition, low responsiveness is negatively perceived by the user. It is worth mentioning that activities that are periodic by essence are sometimes implemented in an aperiodic manner in order to save resources.

Whatever the exact approach, one of the main steps is to derive a model of the arrival patterns for aperiodic activities, what will be called in the following the aperiodic Work Arrival Function (WAF). Then, this aperiodic WAF has to be integrated into the response time analysis. There are however difficulties:

- obtaining aperiodic data (i.e., by measurements or simulation),
- modeling aperiodic data,
- integrating the model into schedulability analysis.

What we are discussing in this chapter is not how to obtain data but how to model it and integrate it into schedulability analysis.

2.1.2 Handling aperiodic traffic

There are two classical approaches to handle the aperiodic traffic:

- worst-case deterministic approach:aperiodic frames are considered as periodic frames with their periods equal to the minimum inter-arrival times, this is the

[1]Because of the incremental design process, in-house usages or constraints of the cooperation process between car-makers and suppliers, priorities on the CAN bus do not necessarily reflect the criticality of the frames (i.e., importance from a functional point of view, deadline constraint).

well known sporadic model [Spuri 1996]. However, in many cases, the minimum inter-arrival time is so small that the resulting workload is unrealistic, and often greater than 100% [Zhang 2008].

- An average-case probabilistic approach: the aperiodic traffic is modeled according to a probabilistic inter-arrivals process, the next step is then to estimate the 'probable' number of arrivals in a given interval of time. This approach is clearly not suited to real-time systems because it largely underestimates the arrivals of aperiodic traffic which can occur in small time intervals [2]

A basic probabilistic framework was set for inclusion of aperiodic frames in a controlled manner using a threshold value in [Burns 2003]. This chapter builds upon this framework and discusses precisely the mechanism of deriving the aperiodic WAF, as well as it removes some assumptions placed in [Burns 2003]. In particular, we show that in our specific context it is not necessary that the different streams of aperiodic frames are modeled individually.

Overview of approach

We do not assume any prior knowledge of the aperiodic frame activation pattern, however we assume that it is possible to monitor the system, or a simulation model of it, and gather data about the arrival times of aperiodic frames. Then, from the measurements, we build a probabilistic model of the aperiodic inter-arrival times under the form of an empirical frequency histogram or a distribution obeying a closed-form equation whenever possible. The next step is to derive a deterministic WAFs from the probability distribution of the aperiodic frame inter-arrival times. A general mechanism is provided enabling to derive the deterministic WAF from the underlying probabilistic distributions of the aperiodic traffic even given in form of empirical histograms, which is worthy in practice since aperiodic arrivals do not necessarily obey a closed-form equation. Another advantage is that the technique is independent of the scheduling and can be used whatever the policy (preemptive, non-preemptive, fixed priority, dynamic-priority, etc) and whatever the task model. All in all, we believe that our proposal offers a better solution for taking into account aperiodic traffic in systems with dependability constraints, compared to worst-case and average case probabilistic approaches.

2.2 System Model

The trace of aperiodic events is characterized by a set $D = E_1, E_2, ..., E_n$ where E_i is an i^{th} aperiodic event such that E_1 is recorded before E_2 on the bus. The events in D are recorded in orders of their arrivals on the bus. Each aperiodic

[2] According to the principle of large deviations: the smaller the interval, the larger (in proportion) the deviation to the mean [Navet 2007].

a	ρ	C (msec)
0.5	341	0.760
0.5	878	0.696
0.5	2000	0.760
9	33	0.632
12	256	0.632

(a) Approximated trace

a	ρ	C (msec)
0.500	341	0.760
1.250	878	0.696
1.954	2000	0.760
9	33	0.632
12	256	0.632

(b) Actual trace1

a'	ρ'	C' (msec)
0.5	341	0.760
1.260	878	0.696
1.956	2000	0.760
9	33	0.632
12	256	0.632

(c) Actual trace2

Figure 2.1: Approximated trace against trace1 and trace 2

event is characterized by a set $E_i = \{a_i, \rho_i, C_i\}$ where a_i is an arrival time ($a_i^{'}$ is the estimated arrival time), ρ_i is a priority of the aperiodic frame and, C_i is the worst-case execution time of the frame. The length of set D depends on the time when trace capture was stopped, but it should be sufficiently large to deduce the probabilistic model of inter-arrivals.

2.3 Modeling aperiodic traffic

The data used in this work comes from measurements taken on-board of a PSA vehicle but because of confidentiality reasons we have obscured the characteristics which could reflect about the design at PSA Peugeot Citröen.

What was measured are the times at which the frames started to be transmitted and not the times at which the transmission requests were issued. Especially when the network is loaded, the two can be significantly different because of frames transmissions being delayed by higher priority frames. This could be taken into account by studying the busy periods on the bus and constructing a worst-case activation process, and is being discussed in section 2.3.1.

2.3.1 Approximating arrival process

The modeling process of the aperiodic traffic involves estimating the probabilistic distribution of aperiodic inter-arrivals from the captured data trace of a simulation model of a vehicle or from a real vehicle. The captured data trace of bus activity gives us the arrival times of frames on the bus, priorities of frames and size of the frames. The difficulty in using this captured data trace lies in the fact that the measured arrival time of the frames on the bus may not coincide with the actual release times

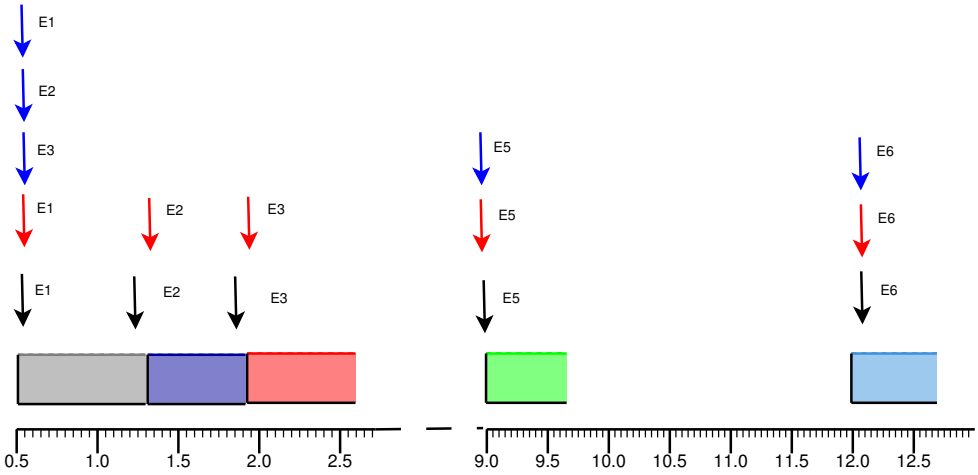


Figure 2.2: Gant chart for trace1: black arrows are actual release times and red arrows are observed arrival times in data trace.The blue arrows will be the approximated arrival times.

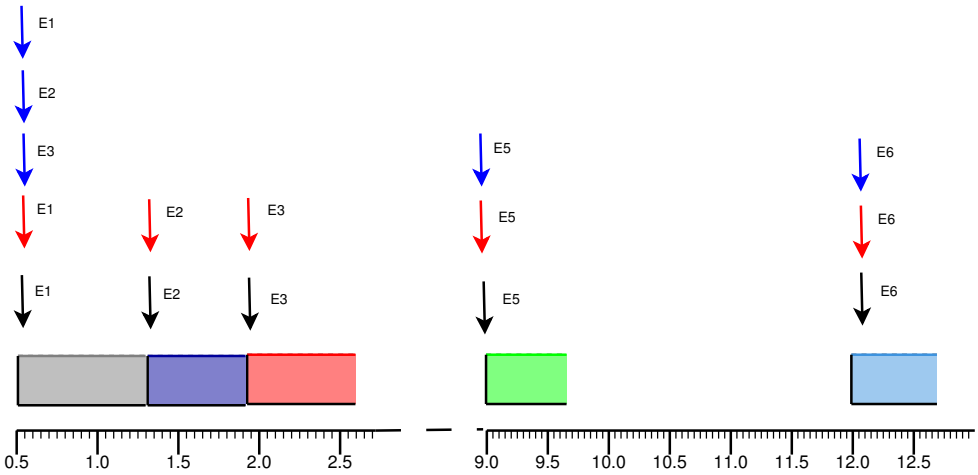


Figure 2.3: Gant chart for trace2: black arrows are actual release times and red arrows are observed arrival times in data trace. The blue arrows will be the approximated arrival times.

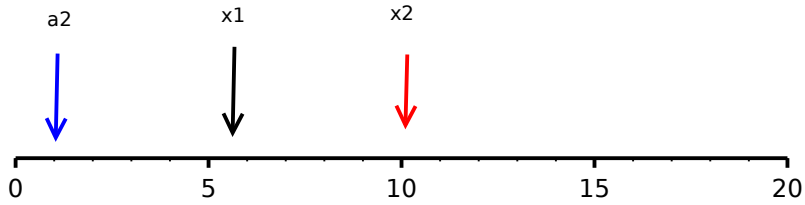


Figure 2.4: Approximation error when approximating the arrival of a frame. The frame arrives at time x_1, observed at arrival time x_2 in data trace and approximated arrival time is at a_2.

of the frames. This requires us to approximate an actual arrival process from the captured data trace. The actual arrival time for some frame i can be approximated by subtracting the level-i busy period seen by the frame. The level-i busy period seen by frame i on bus can be easily computed from a trace. The simple subtraction of the level-i busy period give us the worst-case arrival process of the aperiodic frames, which is what is required. The approximated arrival process for the aperiodic frames gives us the worst-case arrival process which can lead to burstiness in lower priority frames as they are the ones which are pushed back when the aperiodic traffic arrives.

Assumption:

- No inter-frame sequence for frame separation. Otherwise all frames after first frame will be equally shifted by three bit time.
- The data trace is sorted according to arrival times then priorities; such that if two frames arrive at same time then highest priority frame will precede the lower one in the table, which is natural for a captured data trace.

Therefore, for some frame i the level-i busy period seen by it will be equal to the summation of transmission time of all higher priority frames preceding i^{th} frame in data trace; see algorithm 1.

2.3.2 Errors in approximation

When approximating the arrival process from captured data trace e.g. arrival times of table 2.1 we will have an approximation error for the approximated arrival process if the actual arrival process was not the worst-case arrival process e.g. for the trace of figure 2.3 we will get an approximation error as blue and black arrows do not coincide.
Suppose that an aperiodic event occurs at time x_1 and bus is busy transmitting the frames of higher priority. When the level-i busy period for frame released at time x_1 is over it begins transmitting at time x_2 which is observed and recorded in a data trace. When the approximating the time actual arrival time (x_1) of frame from the observed arrival time from trace (x_2) we get a wost-case arrival time of a_2 for the frame which is earlier than x_1 and thus we have an error in the approximation. The

approximation error ϵ is given by: $\epsilon = |x_1 - a_2|$ and is directly dependent upon the length of busy period seen by the frame as $a_2 = x_2 - l$, where l is the length of level-i busy period. The maximum approximation error will occur when the frame arrives near observed arrival time from trace ($x_2 - x_1 \approx 0$) and therefore maximum approximation error is $\epsilon = |x_2 - l|$.

However, we are not concerned by this approximation error as we are interested in the worst-case arrival process.

Algorithm 1 Algorithm for estimation of worst-case arrival time for frame arriving at a_i from captured data trace.

```
while(!EndOfTrace)
    k = i − 1
    j = i
    while(ρ_i > ρ_k && k > 0)
        if(a_k + C_k == a_j)
            continue
        else
            a_i' = a_j
            break
        end
        j = k
        k = k − 1
    end
    if(k > 0)
     a_i' = a_k
    else
     a_i' = a_i
    end
end
```

2.3.3 Finding distribution

In order to model the inter-arrival times of the aperiodic traffic, we first analyze some important structural properties of the data (e.g., linear and non-linear correlation) then find out the probability distribution that best fits our data. The presence of linear and non-linear dependencies in the data would impact its modeling because it would imply a departure from the i.i.d. property (independent and identically distribution). To test these two kind of dependencies, as classically done in exploratory data analysis, we make use of some visual confirmatory tests, the "run sequence plot" and "lag plot" here, as well as the auto-correlation and BDS test (Brock, Dechert, Scheinkman, see[Broock 1996]).

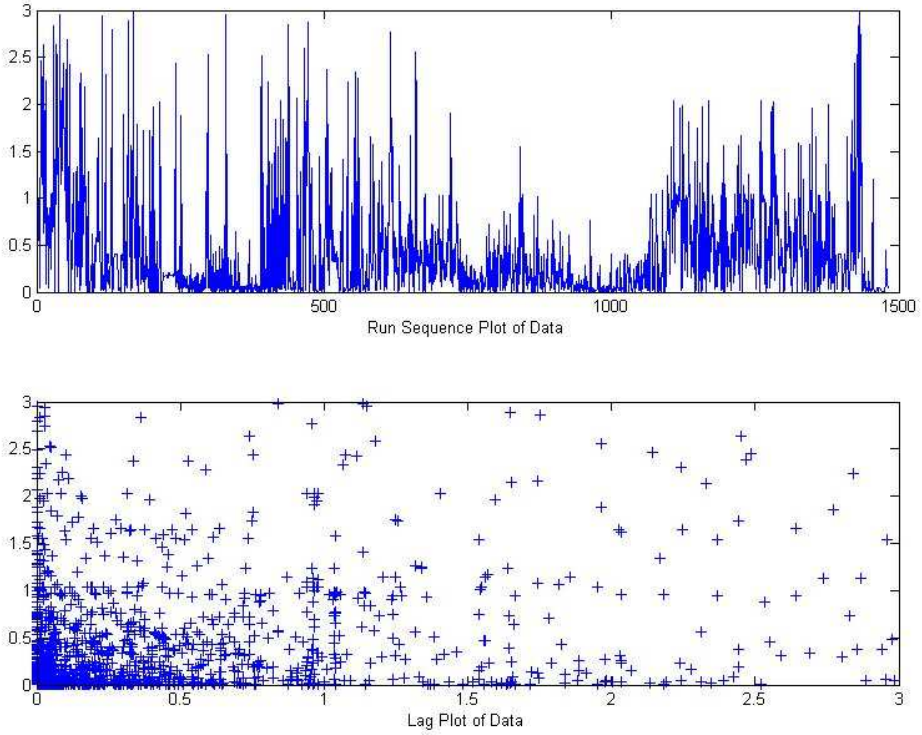


Figure 2.5: Visual analysis of captured data trace. The upper graphic is a run sequence plot where the x-axis is the index of the data points and the y-axis is the time till the next aperiodic arrival expressed in seconds. In the lower graphics, a lag plot, both axes indicates the time till the next aperiodic arrival in seconds.

Run sequence plot

The run sequence plot displays an observed univariate data in a time sequence. It helps to detect outliers and shifts in the process. Figure 2.5(upper) is a run sequence plot of our data trace where the data points are indexed by their order of occurrence. The plot indicates that data does not have any long term shifts in heights over time.

Lag plot

A lag plot helps to gain some insight into whether a data set or time series is random or not. Random data should not exhibit any visually identifiable structure in the lag plot. Figure 2.5(lower) is a lag plot of our data trace (here the lag is chosen equal to 1: $x = X_{k+1}$ and $y = X_k$, where X_k is the k^{th} observation). Since the lag plot appears to be structureless, the randomness assumption cannot be rejected.

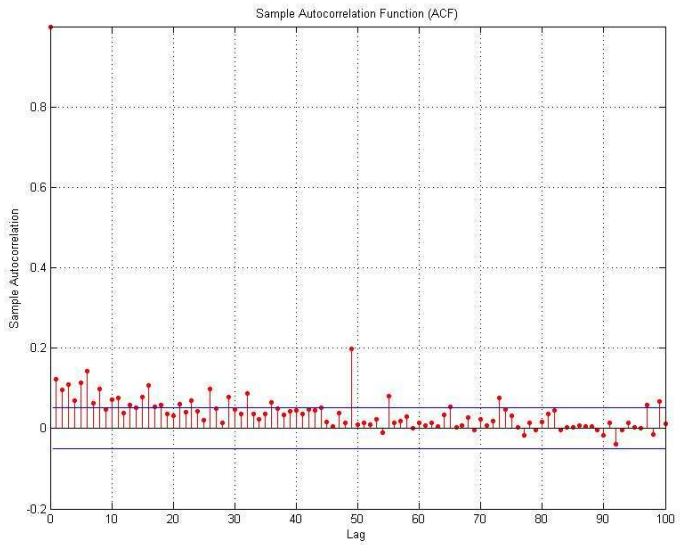


Figure 2.6: Auto-correlation of captured data trace.

2.3.3.1 Autocorrelation analysis

The autocorrelation analysis detects the existence of serial correlations in a data trace. Precisely the correlation of order k indicates the linear relationship that may exist between data values separated by k positions. The first 100 correlation coefficients of the data trace are shown in figure 2.6 associated with the thresholds beyond which the values are statistically significant (1% significance level here). The graphic visualization of the correlation coefficients makes it possible to evaluate the importance and the duration of the temporal dependencies. Here, serial correlations in the aperiodic traffic are relatively limited:

- limited in frequency: on the entire aperiodic traffic, there are only 19 significant auto-correlations coefficients until a lag of 100,
- limited in intensity: the few significant auto-correlations are below 0.2 which is insufficient to be used at ends of predictions.

These autocorrelations can probably be explained by the fact that the activation of certain functions of the vehicle requires the transmission of several consecutive frames, but, the instants of activations of the functions have small correlations. Also, the spike that can be observed around the lag 50 is likely due to a periodic frame that has not been properly filtered out in the data trace.

2.3.3.2 BDS analysis

Auto-correlation has the limitation that it can only test the linear dependency in the data. In order to test for non-linear dependencies a more general statistical test

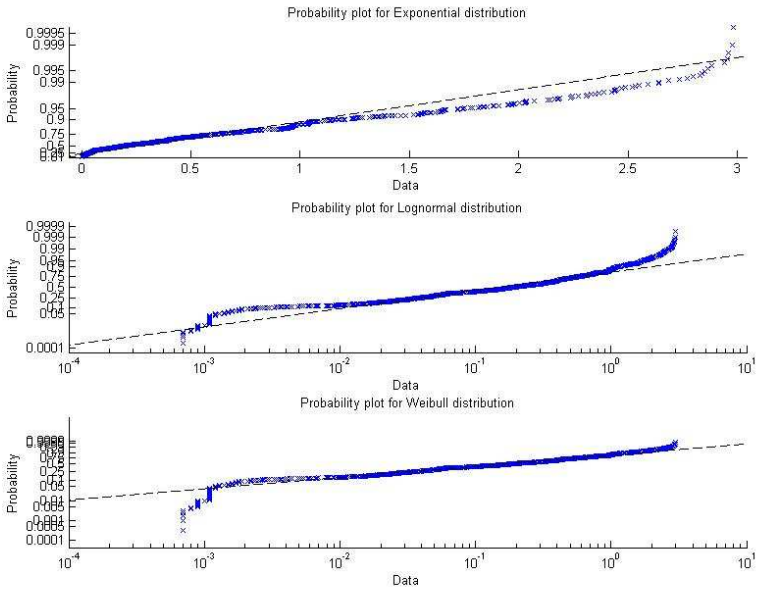


Figure 2.7: Probability plots for 3 candidate distributions, from top to bottom, the exponential law, the log-normal law and the Weibull Law.

than the auto-correlation must be used. One such test is the BDS test [Broock 1996] which employs the concept of spatial correlation from chaos theory to test the hypothesis that the values of a sequence, in this chapter inter-arrival times, are independent and identically distributed (i.i.d.). Deviation from the i.i.d. case will be caused by the non-stationarity of the process (e.g., existence of trends), or the fact that there are linear or non-linear dependencies in the data.

We carried out the BDS test for various combinations of its parameters m and δ (for example for $m = 2$ and $\delta = 3$ as recommended by the authors of the test. For certain combinations we could not reject the hypothesis that the data points are i.i.d. at the 1% confidence level. The results of auto-correlation analysis and BDS test enable us to conclude that it is possible in our specific context to model the aperiodic inter-arrival traffic by a random variable obeying a memory-less probabilistic distribution without diverging from reality.

2.3.3.3 Distribution fitting

We now need to find the probability distribution and its parameters which models the experimental data the most accurately. After having drawn aside certain possibilities for obvious reasons (for example, the normal law because its density function of density is not monotonously decreasing), we tested distributions identified by adjusting their parameters according to the principle of the maximum of likelihood (MLE). Specifically, we have successively considered the exponential law, the log-normal law and the Weibull law. The exponential law was plausible a priori

taking into account the decrease of the density which one can observe in the data trace, the two other laws have been chosen for their well-known flexibility.

2.3.3.4 Probability plots for visual selection

The distribution of the observed data is plotted against a theoretical distribution in such a way that the points should form approximately a straight line. Departures from this straight line indicate departures from the specified distribution. If the probability plot is approximately linear, the underlying distribution is close to the theoretical distribution. What can be observed in figure 2.7 is that the Weibull law is the distribution that best fits the data. This visual conclusion is confirmed by statistical acceptance tests discussed in the next paragraph.

2.3.3.5 Acceptance test

In previous section evaluation of the quality of results was done visually. In this section we use the statistical tests to verify the assumption that data trace follow a particular distribution. Specifically, we are using the χ^2 and Kolmogorov-Smirnov "goodness-off-fit" tests" [Millard 1967, Brumback 1987]. The best results were obtained using the Weibull law, followed at some distance by the log-normal law. The conclusion of the two tests is that one cannot reject the assumption that the data follows a Weibull distribution at a significance level of 1%. For a broad data sample collected on a real system, and not artificially generated data, it is a conclusive result.

Figure 2.8 presents the real data trace and an "artificial" trace generated by a Weibull law with MLE-fitted parameters. It is observed that some "patterns" present in the real trace disappear and that the simulated trace is more homogeneous in time, but overall adequacy of the modeling seems good. From the analysis, carried out in this section, we can conclude that in our specific context the Weibull distribution provides a satisfactory model for the aperiodic traffic inter-arrival times, followed by log-normal and exponential distributions at some distance.

2.3.3.6 Using two parameter distributions

The choice of a distribution is often dictated by the nature of the empirical data which is often over-dispersed and heterogeneous in practice. The selection of a distribution from the family of distributions which are likely to model the empirical data is often governed by the flexibility of the distribution to handle dispersion and heterogeneity. For example the Poisson and exponential distributions are single parameter distribution which implicitly assume simple parametric models and lack in the freedom to adjust the variance independent of the mean, bringing in the handicap to model the dispersed data. A model with additional parameter to take care of dispersion independent of mean may provide a better fit. The weibull and gamma distributions are two parameter distributions which have this flexibility of handling variance independent of mean. Besides these two-parameter distributions

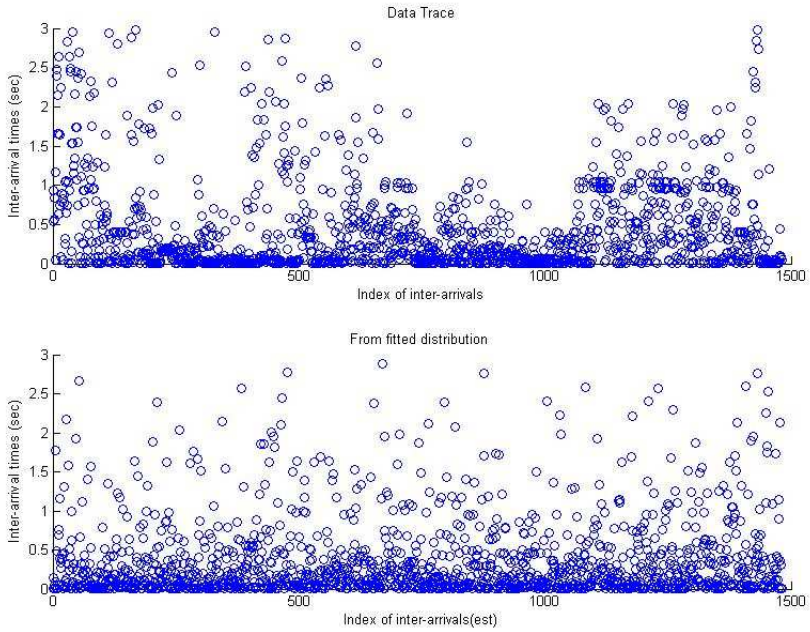


Figure 2.8: Comparison between the captured data trace and a random trace generated by a Weibull model with MLE-fitted parameters.

will converge to the simple parametric distribution depending on the values of the parameters used. For these reason in rest of the work weibull distribution will be used.

2.3.4 Threshold based work-arrival function

$S(t)$ is the aperiodic work arrival function which gives us the number of aperiodic frames in a time interval t and that will be used in the response time analysis. $S(t)$ is an increasing "staircase" function such that the "jumps" in the function correspond to the arrival of an aperiodic frame. To construct this function, we propose to discretize the time and calculate the value taken by $S(t)$ for each value of t between 1 and n where n, expressed in milliseconds, is the largest value that we may reasonably require during the computation of a response time. For example, one can set $n = 1000$ms if the largest period of activity on the bus (*i.e.,* the largest busy period) does not exceed a second.

2.3.4.1 Safety threshold α for $S(t)$

We denote by $X(t)$ the stochastic process which counts the number of aperiodic frames in time interval t. For example, in the data trace which we studied in the preceding sections, inter-arrivals would be controlled by a Weibull law. The idea is to find the "smallest" $\hat{S}(t)$ such that the probability of $X(t)$ introducing aperiodic

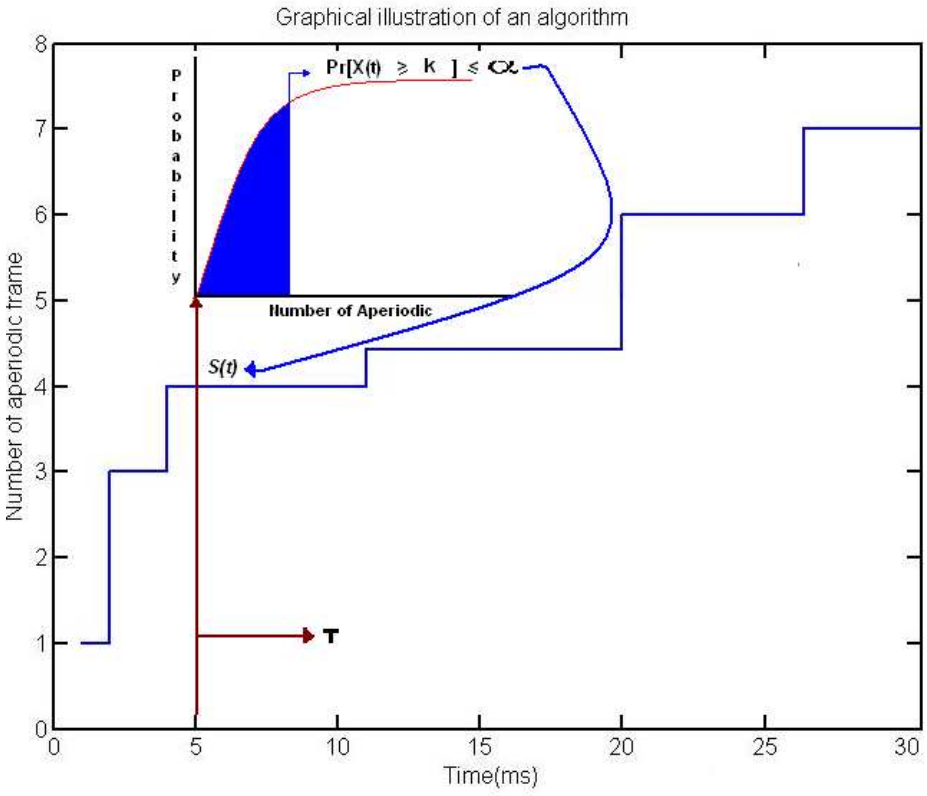


Figure 2.9: Graphical representation of algorithm for computation of $S(5)$. It consists in finding the smallest value of k using the CDF of the inter-arrival distribution according to equations 2.1 and 2.2.

frames equal to n is lower than a threshold value α fixed by the designer. where n is the number of aperiodic frames introduced by $S(t)$. Formally, we are looking for:

$$\hat{S}(t) = \min\{S(t) \,|\, Pr[X(t) \geq n] \leq \alpha\} \tag{2.1}$$

For example, if one sets $\alpha = 0.01$ it means that in no more than 1% of its trajectories the stochastic process $X(t)$ induces more aperiodic traffic than $\hat{S}(t)$. If $X(t)$ models the real aperiodic traffic accurately, the number of aperiodic frames integrated in the calculation of the response time of a periodic frame will have more than 99 percent chances to be higher than what each instance of the frame will undergo. Of course, the choice of α depends on the dependability objectives of (SIL, System Integrity Level, for example) but $\alpha = 10^{-4}$ seems a reasonable value in the context of a body network that will be considered in the experiments hereafter.

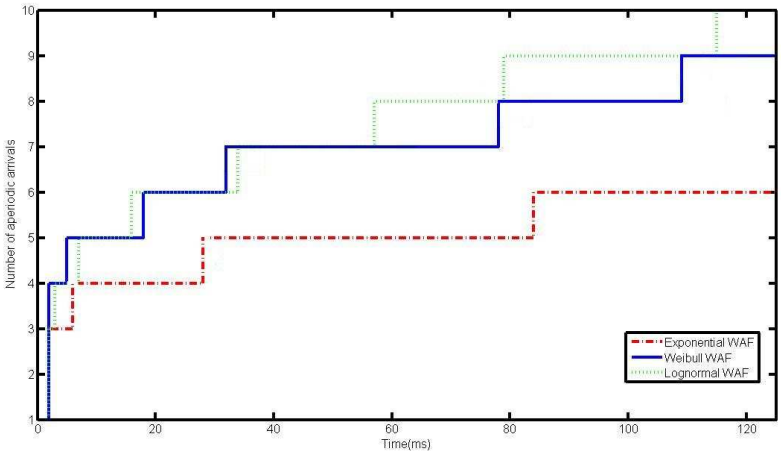


Figure 2.10: WAF using monte-carlo simulations

2.3.4.2 Computation of $S(t)$

We need a way to evaluate $Pr[X(t) = n] \leq \alpha$ at each time instant t. Let $F_n(t)$ be the Cumulative Distribution Function (CDF) of interarrivals.

$$Pr[X(t) = n] = Pr[X(t) \geq n] - Pr[X(t) \geq n + 1] \tag{2.2}$$

$$Pr[X(t) = n] = F_n(t) - F_{n+1}(t)$$

Two cases arise:

- Distribution for which we have a closed-form expressions and can evaluate $Pr[X(t) = n]$ e.g poisson distribution.
- Distribution for which we have no closed-form expression e.g. weibull distribution.

The first case is easy to evaluate using closed-form expression and for the second case we could either resort to numerical or simulation methods to evaluate the equation 2.1.

2.3.4.3 Graphical illustration

Figure 2.9 illustrates the computation of $S(t)$ for a specific value of t, here $t = 5$:

$$\hat{S}(5) = \min\{S(5) \,|\, Pr[X(5) \geq n] \leq \alpha\} \tag{2.3}$$

The probability $Pr[X(5) \geq n]$ can be found using values of $n = 1, 2, 3, ...$ and for $t = 5$ in equation and terminating when probability is more than α.

2.3.4.4 Monte-Carlo simulation approach

We do not always have a discrete distribution modeling the data nor a continuous distribution such that equation 2.1 can be evaluated analytically. We need an alternate method to evaluate equation 2.2 in such cases. This can be done with numerical integration techniques or using Monte Carlo simulation method. The latter approach is described in algorithm 2 where α is the safety level, Δ is the discrete time step, θ is the set of parameters of the aperiodic frame arrival distribution, T is the time horizon, N is the number of random samples[3]to be drawn for the Monte-Carlo simulation. Basically, $S(t)$ is computed for each time unit by drawing N values from the probabilistic distribution modeling the aperiodic frame arrival process and checking if the accumulated probability value smaller than the probability value for which we are evaluating $S(t)$.

Algorithm 2 Deriving $S(t)$ by Monte-Carlo simulation.

Input:$\{T, \alpha, \Delta, \theta, N\}$
Output:$\{S(t)$: The work arrival function$\}$
index = 0;
Data=**random**(θ, N);
for$\{IDX \in 0 : \Delta : T\}$
    for$\{i \in 1 : N\}$
        AccTime = 0;
        k = 0;
        While $\{AccTime<IDX\}$
            AccTime = AccTime+Data[index];
            index=index+1;
            k = k+1;
        end
    end
end
$S(IDX) = k$;

As an illustration of the approach, we derived $S(t)$ in the cases where the aperiodic inter-arrival distribution obeys 1) an exponential law 2) a Weibull law 3) a log-normal law. The number of random draws of the Monte-Carlo simulations (parameter N in algorithm 2) is set to 5 million for each distribution. For all three distributions, the parameters are fitted using MLE against the data traces and the three distributions lead to the same average intensity. What can be observed is that the distribution, and not only the average intensity of the aperiodic traffic, plays a major role in the shape and height of the aperiodic WAF, see figure 2.10.

[3]Central Limit Theorem tells us that the convergence rate is of order $N^{1/2}$ where N is the number of random draws, which means that adding one significant digit requires increasing N by a factor 100. The value of N should be set depending on the threshold α and accuracy objectives.

2.3.4.5 Numerical approach

The WAF is a monotonically increasing staircase curve which returns the number of aperiodic events that have occurred in an interval of time measured from the origin, also know as count model. Let $X(t)$ denote the number of events that have occurred up until time t, $X(t)|t > 0$. Let I_n be the time from the origin to the measurement point where n^{th} event occurred. The relationship between inter-arrival times I_n and the number of events $X(t)$ is :

$$I_n \leq t \;\; \Leftrightarrow \;\; X(t) \geq n$$

We can restate this relationship by saying that the amount of time at which the n^{th} event occurred from the time origin is less than or equal to t if and only if the number of events that have occurred by time t is greater than or equal to n. Therefore, following relationship allows us to derive the count model $C_n(t)$, which returns the number of aperiodic events that have occurred in an interval of time measured from the origin:

$$Cn(t) \;\; = \;\; Pr[X(t) = n] = Pr[X(t) >= n] - Pr[X(t) >= n+1]$$

$$\Longrightarrow \; C_n(t) = Pr[In <= t] - Pr[In + 1 <= t]$$

If we let the cumulative density function (cdf) of I_n be $F_n(t)$, then $C_n(t) = P[X(t) = n] = F_n(t) - F_{n+1}(t)$. In the case where the measurement time origin (and thus the counting process) coincides with the occurrence of an event, then $F_n(t)$ is simply the n-fold convolution of the common inter-arrival time distribution which may (e.g. poisson distribution) or may not (e.g. weibull distribution) have a closed-form solution. For the distributions[4] which do not have a closed-form we can get a closed-form approximation using monte-carlo simulation [Khan 2009] or use a polynomial expansion of $F(t)$ e.g. for weibull distribution we have [McShane 2008]:

$$P[X(t) = n] = C_n(t) = \sum_{j=n}^{\infty} \frac{(-1)^{j+n}(\lambda t^c)^j \alpha_j^n}{\Gamma(cj+1)} \quad n = 0, 1, 2... \tag{2.4}$$

where

$$\alpha_j^0 = \frac{\Gamma(cj+1)}{\Gamma(j+1)} \; j = 0, 1, 2, \ldots$$

and

$$\alpha_j^{n+1} = \sum_{m=n}^{j-1} \frac{\alpha_m^n \Gamma(cj - cm + 1)}{\Gamma(j - m + 1)} \; n = 0, 1, 2, \ldots \;\; j = n+1, n+2, n+3, \ldots$$

[4] Most likely distribution for aperiodic arrivals are exponential, weibull and gamma. And count models for all are available weibull and gamma distribution are of particular interest for their two parameter flexibility. Particularly gamma as the computation of mean and variance is easier in its case as compared to weibull.

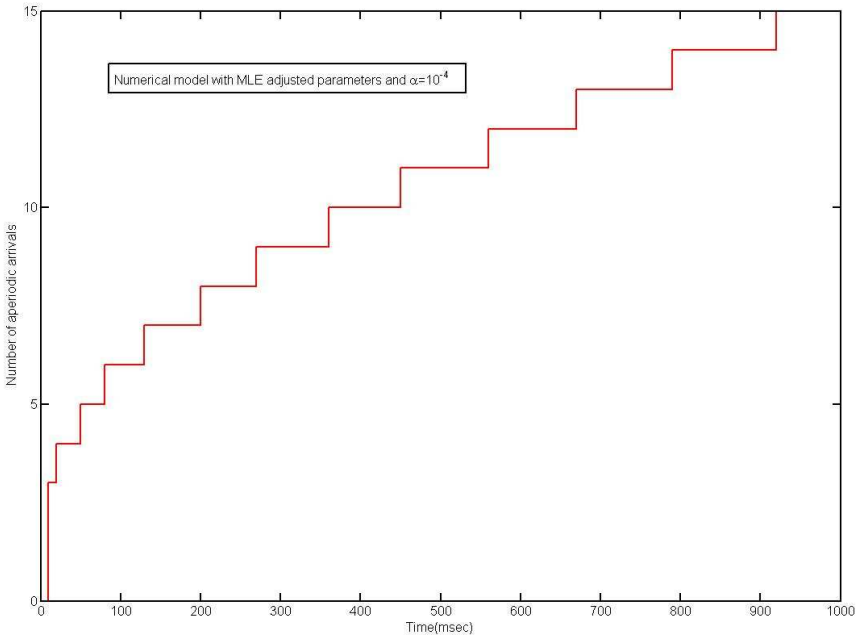


Figure 2.11: Numerical WAF with MLE adjusted parameters and $\alpha = 10^{-4}$

Where the Gamma function is an extension of the factorial function to the real and complex numbers. To build the arrival curves we wish to minimize the probability number of events occurring in an interval in a parametric manner (safety level) for weibull distribution we use equation 2.4 with MLE adjusted parameters, see figure 2.11, such that:

$$S(t) = \min\{Pr[X(t) = n] \leq \alpha\}$$

2.3.4.6 Parameter estimation without data trace

Because of cost and design time constraints, it is not always possible to derive the inter-arrival model from a real data trace, or traces of simulation. This is often the case in automobile projects. In such a situation, as an approximation, a solution is to set the parameters of the distribution based on already known parameters corresponding to another electronic architectures. In the following, we show how to adapt a Weibull[5] model to a new intensity of the aperiodic traffic.

The expected value of a random variable obeying a Weibull law is:

$$E(X) = \lambda\Gamma(1 + \frac{1}{k}) \tag{2.5}$$

[5]The case of single parameter distribution such as the exponential law is trivial, a similar approach can be used for the log-normal law.

where λ is the scale parameter, k is the shape parameter of the Weibull law and the Gamma function is an extension of the factorial function to the real and complex numbers. There exist many, more or less precise, approximations to calculate the gamma function. One good approximation is given by the following formula:

$$\Gamma(z) \approx \sqrt{\frac{2\pi}{z}}(\frac{1}{e}(z + \frac{1}{12z - \frac{1}{10z}}))^2 \tag{2.6}$$

To adjust the expected value of the Weibull law for a new vehicle project, one simply has to change the scale parameter λ to the targeted intensity of the aperiodic traffic. The larger the scale parameter, the more spread out the distribution is i.e. if λ is large, then the distribution will be more spread out and if λ is small then it will be more concentrated. The shape parameter k simply affects the shape of a distribution and is independent of other distribution parameters. In first approximation, we assume here that the shape of the distribution should not change very importantly from project to project and so set the parameter k. This assumption should be verified in the light of the analysis of additional data traces but this is left as a future work. The network load of the aperiodic traffic, denoted ρ, obeys the relation:

$$\rho = (\frac{1}{E(X)}).\bar{A} \tag{2.7}$$

where $\bar{A}$ is the average transmission time of an aperiodic frame. From equations 2.5, 2.6 and 2.7, one obtains:

$$\lambda = (\frac{1}{\Gamma(1 + \frac{1}{k}).\rho}).\bar{A} \tag{2.8}$$

By replacing the values of network load, ρ, and average transmission time, $\bar{A}$, by the values which correspond to the automotive network that one wants to model, one obtains the new value of λ.

2.3.5 Handling priority

A priority assignment policy assigns a priority ρ_i to each frame. The priority assignment function which maps the priorities to these frames from a finite set of values (e.g. 1-2048) depends on the scheduling algorithm. For example in case of Rate Monotonic (RM) scheduling the priorities are mapped based on the periods. Here, we are considering fixed priority scheduling. In order to integrate correct amount of aperiodic traffic we have to take into account the priorities of arriving frames in a work arrival function. The mechanisms to handle priority in a probabilistic framework have been discussed in subsequent subsection.

2.3.5.1 Modeling each priority level

In order to model each priority level individually we will have to filter set of aperiodic events from trace D into subsets $\hat{D}_i$ such that each subset contains aperiodic events of one priority level only, formally:

$$\hat{D}_i = \{\forall E_j \in D | \rho_j = i\} \tag{2.9}$$

Each $\hat{D}_i$ is used to find the WAF against it, assume $S_i^{\alpha,M}(I)$ is the WAF for $\hat{D}_i$. In order to find the higher priority aperiodic load seen by some frame of priority m we will integrate all WAFs for $\hat{D}_i$'s of higher priority than m as:

$$W_m(I) = \sum_{\forall i \leq m} S_i^{\alpha,M}(I) \tag{2.10}$$

The equation 2.10 returns the number of aperiodic frames of higher priority than m in an interval I.

The solution discussed above is an ideal solution, but in realistic problems we will not have enough data points to correctly model the distributions for each priority level, and thus we will have to look for alternate approximate solutions to this problem.

2.3.5.2 Modeling priority using intensity level

Another approach for modeling priorities in schedulability analysis is model all aperiodic traffic as one distribution and control the intensity of traffic for different priority levels using ρ and then re-estimating the λ parameter using equation 2.8, which controls the scale of the distribution and thus governs the intensity of the aperiodic traffic. The higher priority frames could take into account work-arrival curves with larger ρ and lower priorities frames could take into account work-arrival curves with smaller ρ.

2.3.5.3 Modeling priority using groups

Reusing the notation of subsection 2.3.5.1 let $\hat{D}_i$ be a set such that it contains frame of priorities between $1 \,\&\, i$. Formally:

$$\hat{D}_i = \{\forall E_j \in D | \rho_j \in \{1..i\}\} \tag{2.11}$$

$\hat{D}_i$ from equation 2.11 is then used to find work arrival function for each $\boldsymbol{i}$, i.e. for each priority, using the mechanism discussed in in subsection 2.3.4. In order to find higher priority interference for frame with priority m we will use $\hat{D_m}$ to find WAF which returns the number of frames to integrate into schedulability analysis as:

$$W_m(I) = S_m^{\alpha,M}(I) \tag{2.12}$$

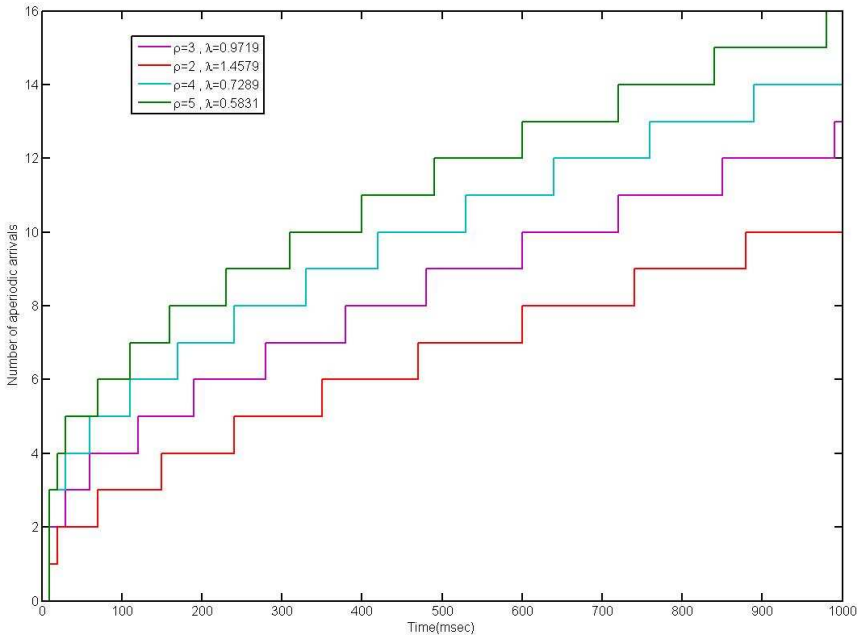


Figure 2.12: Work-arrival curves from weibull distribution for different values of α

The above equation return the number of higher priority frames seen by frame m in an interval I. This seems to be most refined approach among discuss above in terms that it provides intuitive approximation mechanism for integrating aperiodic traffic based on priorities. However it may be susceptible to loss in accuracy for higher priority frames when we do not have enough data points to model the distribution correctly.

2.3.5.4 Comparison of two approaches

This sections presents the comparison between two approaches outlined in subsections 2.3.5.2&2.3.5.3 above. The data trace was filtered to extract various priority groups and then the distribution parameters for each priority group was adjusted using MLE. And for the "intensity level" approach the distribution parameters were found for the whole data trace using MLE and then using equation 2.8 a new intensity parameter was estimated by retaining the value of shape parameter found first time and changing the aperiodic load.

The trends in the work arrival functions of the two approaches is almost same. However, intensity level is introducing more aperiodic work as compared to the priority group approach. The reason for that is when changing the aperiodic load on the network for "intensity level" approach we are basically increasing the intensity parameter of the distribution while retaining the shape of the distribution. Which

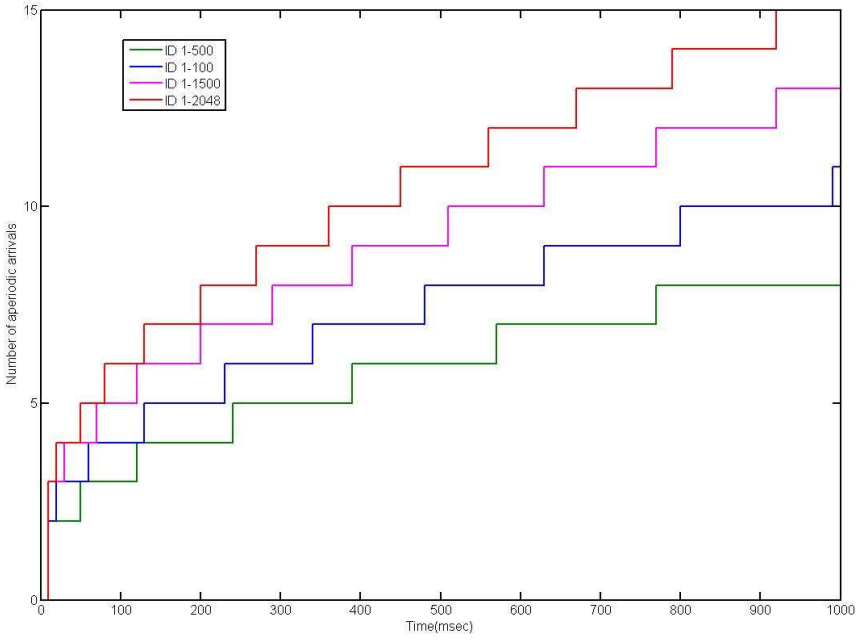


Figure 2.13: Work-arrival curves from weibull distribution for different priority groups

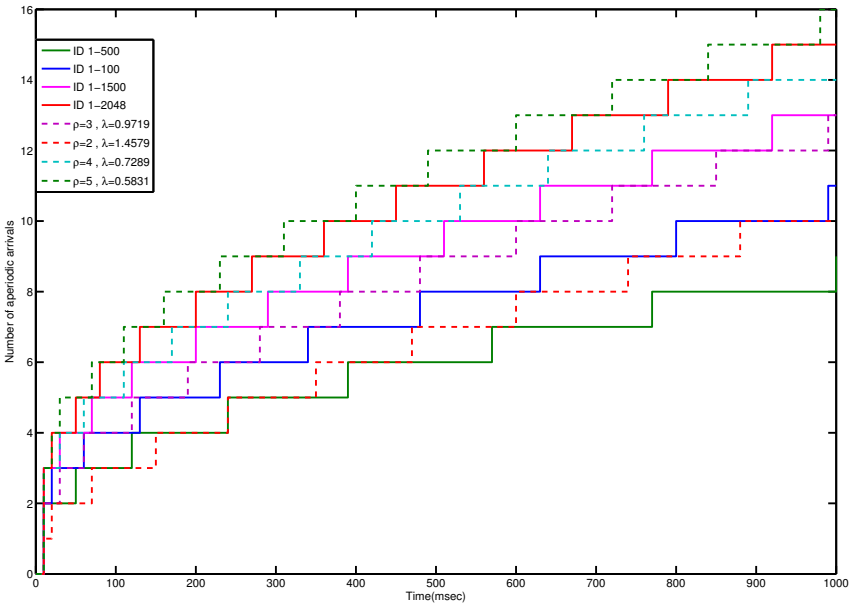


Figure 2.14: Comparison of 'priority group' depicted by solid lines in figure and 'intensity level' depicted by dotted lines in the figure.

essentially means that more traffic is arriving in given interval of time, so for two intensity level one with more aperiodic load will exhibit higher aperiodic traffic than the other level for same interval. The priority group approach is a refined approach, however it may suffer from the lack of data for some priority value.

2.4 Schedulability analysis

Classically, schedulability analysis for real-time communication networks assume periodic or sporadic streams of frames [Tindell 1995, Davis 2007]. In this chapter, for the sake of simplicity, we make use of a sufficient but not necessary schedulability test[6] presented in [Davis 2007] as the framework to include aperiodic WAF into the schedulability analysis. However, the approach would remain similar with the sufficient and necessary test proposed in the aforementioned paper.

In the following, we re-use the concepts and notations from [Davis 2007]. The worst-case response time of frame m is made up of several elements:

1. An upper bound on the queuing jitter J_m,

2. The longest transmission time C_m,

[6] This test is applicable when deadlines do not exceed their periods.

3. The waiting delay w_m at the sending end, that is the longest time that the frame can wait before it starts being successfully transmitted (i.e., before it wins the arbitration on the CAN bus). This delay is given by equation 2.14,

The waiting delay w_m includes the interference due to the aperiodic frames of higher priority than m, which is given by the function $N_m^{\alpha,\mathcal{M}}(t)$ defined as follow:

$$N_m^{\alpha,\mathcal{M}}(t) = S_m^{\alpha,M}(t).\max_{j \in HpAf(m)} C_j \tag{2.13}$$

where $\mathcal{M}$ is the aperiodic interarrival model, α the chosen safety threshold, $S_{\mathcal{M}}^{\alpha}(t)$ the corresponding aperiodic WAF and $HpAf(m)$ is the set of aperiodic frames having higher priority than frame m. It has to be pointed out that the definition of $N_m^{\alpha,\mathcal{M}}(t)$ can use any priority modeling approaches discussed in sections 2.3.5.1 to 2.3.5.3.

As classically done, the waiting delay w_m can be determined with the following recurrence relation:

$$\begin{aligned} w_m^{n+1} &= N_m^{\alpha,\mathcal{M}}(w_m^n) + \max(B_m, C_m) \\ &+ \sum_{\forall k \in hp(m)} \lceil \frac{w_m^n + J_k + \tau_{bit}}{T_k} \rceil C_k \end{aligned} \tag{2.14}$$

where $hp(m)$ is the set of frames with priority higher than m, and $\max(B_m, C_m)$ corresponds to the longest possible time for which an invocation of frame m can be blocked either by lower priority messages or due to the previous invocation of the same frame. The recurrence relation goes on until $J_m + w_m^{n+1} + C_m > D_m$ or $w_m^{n+1} = w_m^n$. In the former case, the frame is not schedulable while in the latter case the worst-case response time of the frame is given by:

$$R_m = J_m + w_m + C_m \tag{2.15}$$

2.5 Case study

In this section, we illustrate the analysis of nine typical 125Kbit/s automotive body networks with. We used *Netcarbench* [Braun 2007], a GPL-licensed software that generates sets of messages according to parameters defined by the user. The characteristic that a user can describe are network load, number of ECUs, distribution of the periods of the frames, etc. The characteristics used to generate test networks were chosen by setting the details listed in table 2.1 for Netcarbench.

The properties of resulting set of networks that were generated are having characteristics as described in the table 2.2. These networks will be used to analyze the effect of aperiodic traffic by integrating the aperiodic WAFs.

(a) The weighted distribution of periods with random priority assignment of priority from the specified range for test network generation.

SNo.	Period(msec)	weight	Priority Range	Margin
1.	50	2	1-200	1
2.	100	15	1-600	3
3.	200	15	1-1000	3
4.	500	30	200-1000	5
5.	1000	25	300-1500	5
6.	2000	5	500-1500	1

(b) The weighted distribution of frame sizes for test network generation.

SNo	Size(bytes)	Weight	Margin
1.	1	1	1
2.	2	1	1
3.	3	1	1
4.	4	1	1
5.	5	2	1
6.	6	2	1
7.	7	2	1
8.	8	8	2

(c) Characteristic of load and ECU range for generating body networks using Netcarbench

SNo.	Parameter	Range
1.	Load	40 to 45
2.	ECUs	15 to 20

(d) Designating loaded ECUs, i.e. the percentage of overall bandwidth sent by a particular ECU

SNo.	ECU ID	Load(%age)
1.	1	30
2.	2	15
3.	3	10

Table 2.1: Characteristics for generating test networks

SNo.	Test case	ECUs	Load(periodic)	frames
1.	Net1	15	44.24%	110
2.	Net2	17	41.42%	120
3.	Net3	16	43.99%	142
4.	Net4	17	42.04%	105
5.	Net5	19	43.68%	120
6.	Net6	19	43.61%	131
7.	Net7	19	41.94%	117
8.	Net8	19	41.97%	115
9.	Net9	19	40.49%	110

Table 2.2: Test networks generated for body networks of a car.

S.No.	Analysis#	Remarks
1.	WCRT0	without any aperiodic traffic
2.	WCRT1	with aperiodic traffic in the priority levels (1-100)
3.	WCRT2	with aperiodic traffic in the priority levels (1-500)
4.	WCRT3	with aperiodic traffic in the priority levels (1-1500)
5.	WCRT4	with aperiodic traffic in the priority levels (1-2048)
6.	WCRT5	with aperiodic traffic in intensity levels (2)
7.	WCRT6	with aperiodic traffic in intensity levels (3)
8.	WCRT7	with aperiodic traffic in intensity levels (4)
9.	WCRT8	with aperiodic traffic in intensity levels (5)

Table 2.3: For each generated network we are going to perform above listed analysis; which have been tuned according to the priority distribution.

The aperiodic WAFs used to test the affect on the worst-case response times of all generated test networks are shown in figures 2.12 and 2.13. The aperiodic WAFs are generated for designated priority ranges and for various aperiodic loads to study the affect of aperiodic frame priorities and of changing aperiodic load on the periodic message sets. The WAFs are generated from the numerical model of Weibull distribution with a safety threshold $\alpha = 10^{-4}$.

The WCRT of the frames are computed with the software *NETCAR-Analyzer* from *RealTime-at-Work* whose purpose is to analyze the performances of CAN-based communication systems and optimize their design and configuration (*e.g.*, choices for the message priorities and offsets, waiting queue policy and length, etc). Each message set was analyzed for all aperiodic arrival curves in figures 2.12 and 2.13. The resulting response times are shown in figure 2.15 (for message set 3 of table 2.2) are againts all arrival curves listed above. Figure 2.16 shows the relative increase, with respect to no aperiodic traffic case, in the worst-case response times of periodic frames for message set 3 in presence of aperiodic frames, for message set 3, according to WAFs listed above. Figure 2.17 shows the relative increase, with respect

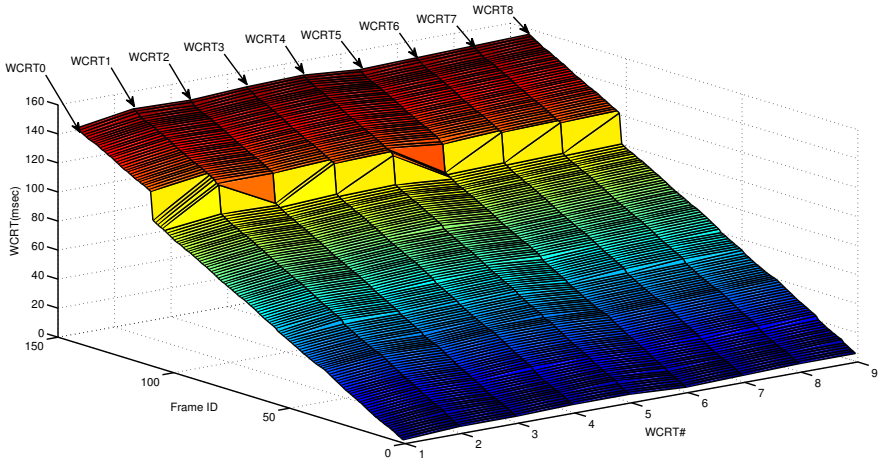


Figure 2.15: WCRT of all cases in the table 2.3 for message set 3.

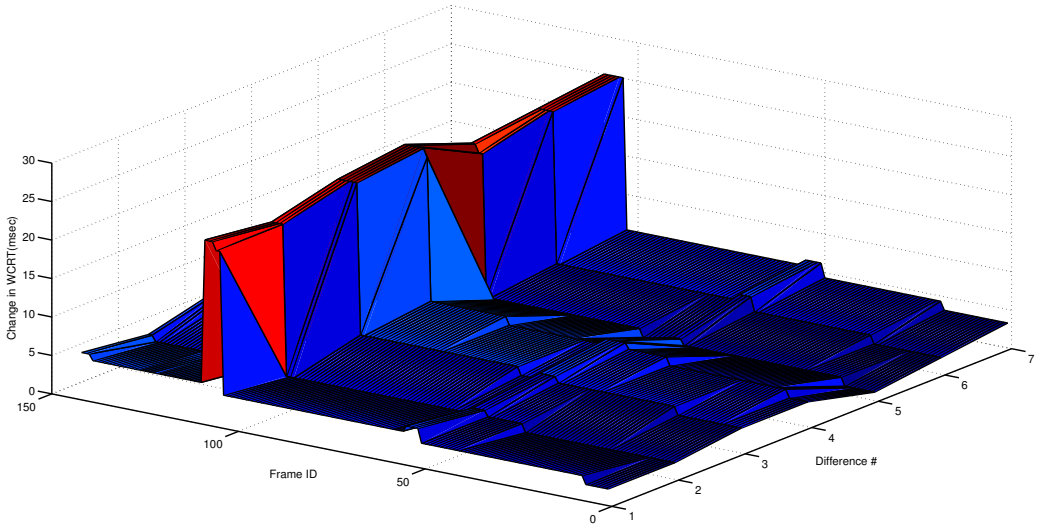


Figure 2.16: Difference between case WCRT0 and other WCRT cases of table 2.3 for message set 3, showing the relative increase in WCRTs with respect to WCRT0.

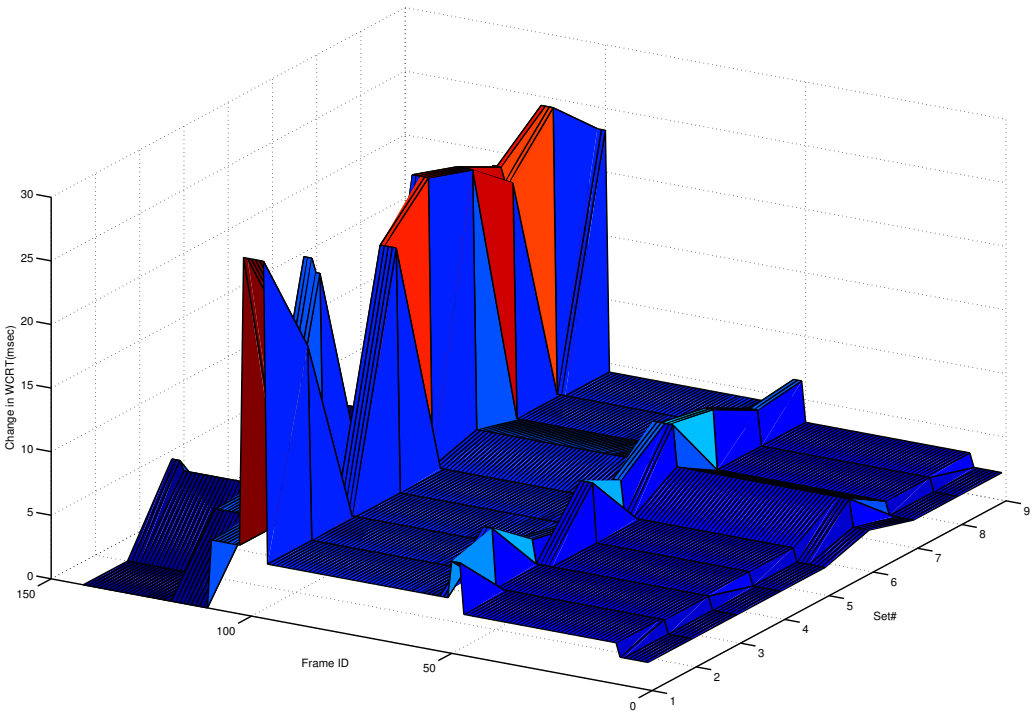


Figure 2.17: Difference between cases WCRT0 and WCRT1 for all message sets, showing the relative increase in the WCRT for all message sets using a fine grained approach.

to no aperiodic traffic case, in the worst-case response times of periodic frames for all message sets using just one work arrival curve (ID=1-500) from figure 2.13.

Even in this context where the periodic load is moderate (e.g. 43.99%) and the aperiodic traffic is limited, one observes that aperiodic traffic rather significantly impacts the worst-case response times of the periodic frames. For instance, the WCRT for the frame with id 107 raises from 98.66ms without aperiodic traffic to 122.7ms with first curve WCRT1 in table 2.3 (+24%). We observe that other WCRT curves also give somewhat similar results. However, the location of aperiodic traffic is different and thus the percentage increase see by frames over experiments may not be same, thus aperiodic traffic plays some role and thus cannot be overlooked. Which can also be verified from the results of other message sets depicted in figure 2.17.

2.6 Summary

In this chapter, we developed a new approach for integrating the aperiodic traffic in response time analysis. The main interest of the proposal is that the overestimation of the aperiodic traffic is kept to the minimum that still enables the system to meet some chosen dependability requirements.

However, the resulting response time estimation can be pessimistic especially for lower priority frames when there is a large volume of aperiodic traffic, as we have assumed worst-case arrival process when estimated the release times from data trace. The estimated arrival process is bursty in nature and will be seen more by the lower priority frames. It is possible to be less pessimistic by modeling each aperiodic stream individually and integrate only the higher priority aperiodic WAFs into the schedulability analysis. However, we believe that this more fine-grained approach will not be always practical since it requires significant modeling efforts and large quantity of data traces. We have provided few schemes which would minimize the pessimism due to priority issues and still respecting the safety threshold while being as accurate as possible (*i.e.*, discard as much as possible of the lower priority aperiodic traffic).

Chapter 3

Schedulability analysis with hardware limitations

Contents

The analysis of the real-time properties of an embedded communication system relies on finding upper bounds on the Worst-Case Response Times (WCRT) of the messages that are to be exchanged among the stations. The classical WCRT analysis of Controller Area Network (CAN) implicitly assumes an infinite number of transmission buffers and negligible copy-time. However, in reality, CAN controller may have some characteristics, such as non-abortable transmissions, which may significantly increase the WCRT. Which, if not considered, may result in a deadline

violation due to an additional delay. In this work, we explain the cause of this additional delay and extend the existing CAN schedulability analysis to integrate it. Finally, we suggest implementation guidelines that minimizes both the run-time CPU overhead and the additional delay due to priority inversion.

3.1 Introduction

Controller Area Network (CAN) was specifically designed for use in the automotive domain and has become a de-facto standard. Today, high-end cars can contain as many as 70 CAN controllers [Navet 2005]. CAN has been extensively used in other areas as well, including industrial automation, especially networked control systems [Marti and 2010], because of its interesting real-time properties and low-cost. Whatever the domain, existing schedulability analyses of real-time applications distributed over CAN assume that:

1. If a CAN node has to send out a stream of messages having the highest priority on the bus, it should be able to do so without releasing the bus between two consecutive messages, despite the arbitration process that takes place at the end of each transmission.

2. If on a CAN node more than one message is ready to be sent, the highest priority message will be sent first. This means that the internal organization and message arbitration of the CAN node is such that this is possible.

These assumptions put some constraints on the architecture of the CAN controllers and on the whole protocol stack. Sometimes, because of the CAN controller or protocol layers, priority inversion among messages can occur. This can happen when the controller sends more distinct messages than the number of transmission buffers available and transmission requests (for low-priority messages) cannot be cancelled. Indeed, some CAN controller hardware implementations have internal organization such that they send messages independent of CAN message ID (Microchip MCP2515, Freescale MC68HC912), send messages in a FIFO order (Infineon XC161CS), or do not have enough transmit buffers (Philips SJA1000). Moreover, the transmit buffers may be managed without abortion (Philips 82C200) [Natale 2006], or the support for abort mechanisms may be missing at the device driver level or, finally, the communication stack may be configured such that it does not support cancelling transmission (see "transmit cancellation" in an AUTOSAR stack, page 37 in [AUTOSAR 2009]). As a result, a message can be delayed for a longer time than is expected by classical analyses [Tindell 1995, Davis 2007] and the response time increases accordingly.

Problem with current analysis

Timing analyses of CAN developed over the years model the network as an infinite priority queue where each node is inserting its messages according to their priority. It is then assumed that the highest priority message in the queue wins the arbitration, be it in the deterministic [Tindell 1995, Davis 2007, Grenier 2008] or stochastic

case [Zeng 2010, Hansson 2002]. However, this model does not hold when hardware and software constraints, like limited numbers of transmission buffers in the CAN controller and copy-time[1] of messages from device drivers, are considered Then the Worst-Case Response Time (WCRT) increases as compared to the traditional analyses. To the best of our knowledge, this issue was first identified and analysed in [Meschi 1996].

Some work has already been carried out to identify and analyse the effects of limited transmission buffers, in [Meschi 1996, Natale 2006, Natale 2008] and [Khan 2010]. In [Natale 2008], Natale classifies and explains all the cases leading to priority inversion due to hardware and software limitations, that were not covered by the existing analyses. In [Meschi 1996] Meschi et al. show that at least three transmission buffers are needed to avoid priority inversions when the copy-time of a message from the queue to the controller is neglected. However, analysis in [Meschi 1996] only addresses the case when transmission requests are abortable. In [Khan 2010], Khan et al. address the case of priority inversion in an abortable CAN controller when copy-time of messages and the architecture of a device driver is taken into account. In [Davis 2011a], Davis et al. provide schedulability analysis when device drivers use FIFO[2] transmission queues. However, the analyses provided in [Khan 2010, Davis 2011a] do not investigate the non-abortable CAN controller case. In [Natale 2006] Natale provides an analysis for integrating the increase in WCRT due to priority inversion in non-abortable CAN controllers. However, the analysis provided in [Natale 2006] takes into account the interference of all lower priority messages for the message which suffers from priority inversion, which may not be the case as is shown in this paper. Furthermore, it does not consider the fact that the increase in the WCRT (additional delay) of a message manifests itself as a jitter for lower priority messages.

Contributions of this work

The effects of a limited number of transmission buffers have been identified in [Tindell 1994c], [Natale 2006] and [Meschi 1996]. In [Natale 2006] the author gives the analysis for the case when it is not possible to cancel transmission and in [Meschi 1996] the authors show that at least 3 transmission buffers are needed to avoid priority inversions when the copying time of a message from the queue to the controller is neglected. Here, we address the 3 or more buffer case with two scenarios. First is the case when it is possible to cancel a transmission request and when the copying overhead can take any reasonable value and the second case is when it is not possible to cancel a transmission request. We derive a worst-case response time analysis that integrates these two cases in this chapter.

[1] This time could be worst-case execution time of an interrupt service routine plus interrupt latency for interrupt based system. For polling based systems it could be worst-case execution time of task putting message in transmission buffer plus polling tick duration.

[2] At least one commercial tool, namely NETCAR-Analyzer from RTaW (see http://www.realtimeatwork.com/?page_id=396) , addresses the FIFO case.

CAN Controller	# Tx Buffers	Priority for transmission
Microchip MCP2515	3	Independent of CAN ID. For example, if two buffers have same priority(*11* is highest and *00* is lowest) settings the buffer with highest buffer number will be sent first. Aborting a frame in a Tx buffer is possible.
Freescale MC68HC912	3	Independent of CAN ID, A 8-bit local priority field is managed by application software. Aborting a frame in a Tx buffer is possible.
Infineon XC161CS	32, (Tx/RX)	Scalable FIFO
Philips SJA1000	1	Aborting a frame in a Tx buffer is possible.

Table 3.1: Characteristics of different CAN controllers.

Besides, we provide guidelines for an optimized CAN driver implementation. The case addressed here is meaningful because in practice most CAN controllers have more than 3 buffers and possess the ability to cancel a transmission request may or may not be supported by them, the device drivers or the higher level communication stack.

These assumptions put some constraints on the architecture of the CAN controllers and on the whole protocol stack. Sometimes, because of the CAN controller or protocol layers, priority inversion among messages do occur. This happens in particular when the controller sends more distinct messages than the number of transmission buffers available and when transmission requests (for low-priority frames) cannot be canceled. Indeed, some CAN controllers do not allow to cancel a transmission request, or the support for abort mechanisms is missing at the device driver level or, finally, because the communication stack does not support it (see "transmit cancellation" in an AUTOSAR stack, page 37 in [AUTOSAR 2009]). As result, a frame can wait for a longer time what is expected by classical analysis [Tindell 1995, Davis 2007] and the response times would increase accordingly.

This work provides tighter bounds on the WCRT by identifying more precisely the interference brought by lower priority frames and it also identifies and integrates the jitter due to this interference in the analysis, which may increase the response times for some frames.

3.2 Working of a CAN controller

The configuration and management of the peripheral transmit and receive objects is of utmost importance in the evaluation of the priority inversion at the adapter and of the worst case blocking times for real-time messages.

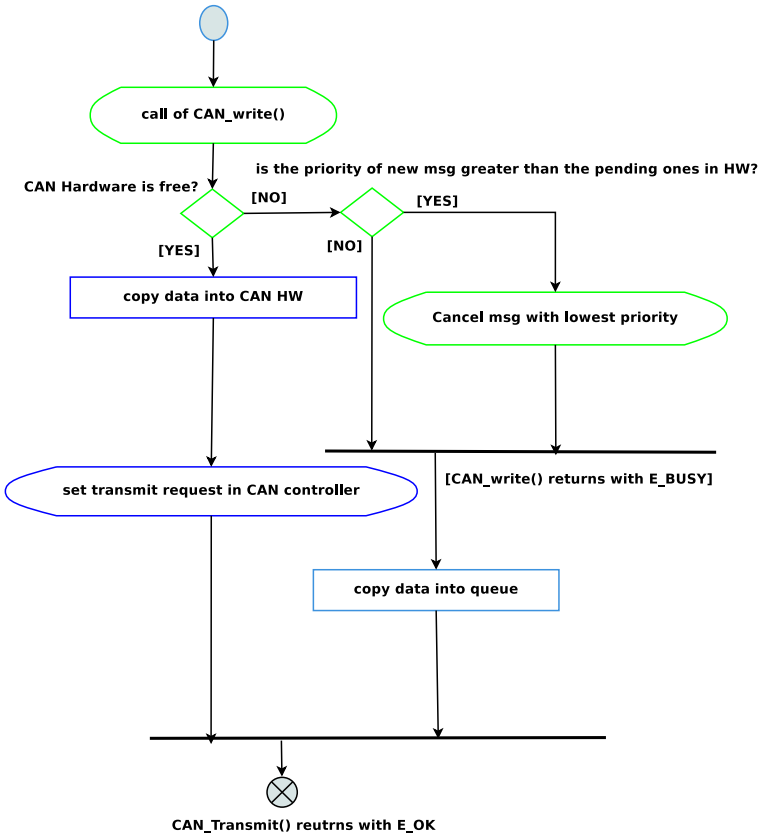


Figure 3.1: AUTOSAR CAN driver message transmit flow.

There is a variation among CAN controllers in terms of architecture for example the variation in terms of number of transmission and reception buffers, flexibility of designating a register as transmission buffer or reception buffer in some CAN controllers. Further, when CAN controller buffers are filled with multiple messages, most CAN controllers select a message for transmission with the lowest identifier, not necessarily the message with the lowest CAN ID. Furthermore, Most CAN controllers, a message that is currently in transmission buffers can be aborted, unless the transmission is actually taking place, see table 3.1 for details.

3.2.1 AUTOSAR CAN driver implementation

The requirement that the highest available message at each node is selected for the next arbitration round on the bus can be satisfied in several ways. The simplest solution is when the CAN controller has enough transmission buffers to accommodate all the outgoing messages. This solution is possible in cases as in some CAN controllers the transmission and reception buffers could be as high as 32 and the CAN device driver can assign ea sh outgoing message a buffer.

However, this is not always possible in current automotive applications where a relatively large number of buffers must be reserved for messages in order to avoid

message loss by overwriting. Furthermore, for some ECUs, the number of outgoing messages load can be very large, such as, for example, gateway ECUs. Besides, in the development of automotive embedded solutions, the selection of the controller chip is not always an option and designers should be ready to deal with all possible HW configurations.

Too overcome these problems solutions exist which give implementation guidelines from device drivers, e.g. AUTOSAR CAN driver specification. To overcome the limited buffer issue these advocate implementing queues in drivers and for preserving the priority order would require that:

- The queue is sorted by message priority (message CAN identifier)
- When a transmission buffer becomes free, the highest priority message in the queue is immediately extracted and copied in place of emptied buffer.
- If, at any time, a new message is placed in the queue, and its priority is higher than the priority of any message in the transmission buffers, then the lowest priority message holding a transmission buffer needs to be aborted, placed back in the queue and the newly en-queued message copied in its place and,
- Messages in the transmission buffers must be sent in order of their CAN identifiers.

The AUTOSAR transmit request API is a common interface for upper layers to send messages on the CAN network, see figure 3.1. The upper communication layers initiate the transmission only via the CAN Interface services without direct access to the CAN driver. The initiated transmit request is successfully completed, if the CAN driver could write the message into the CAN hardware. However, if no transmission buffers were available at the time of initiation, the state of the transmit request obtains the state "pending" and the message is temporarily stored in the CAN Interface. When the previous transmission is completed and transmission buffers are released the subsequent transmit requests are carried out. If no hardware and also no software buffers are available the transmit request is rejected immediately.

All pending transmit requests are transmitted in priority order, implicitly defined by the CAN ID. The abort of pending messages within the transmit buffers is necessary to avoid inner priority inversion. The mechanism of the transmit processing differs, whether hardware cancellation is supported or not. If the hardware cancellation is not supported and the message initiated has higher priority and if all available transmission buffers are busy, this message is delayed until a transmission buffer is released, this may result in a priority inversion.

However, if the transmit cancellation is supported and used (as this can be configured to be TURNED OFF in AUTOSAR) at time of a new transmit request the CAN driver checks for the availability of the transmission buffer. If all buffers are in use, the CAN ID of the requested message transmission is compared with the CAN ID of all pending messages in the transmission buffers of CAN controller. If the requested message transmission has a higher priority compared to the pending

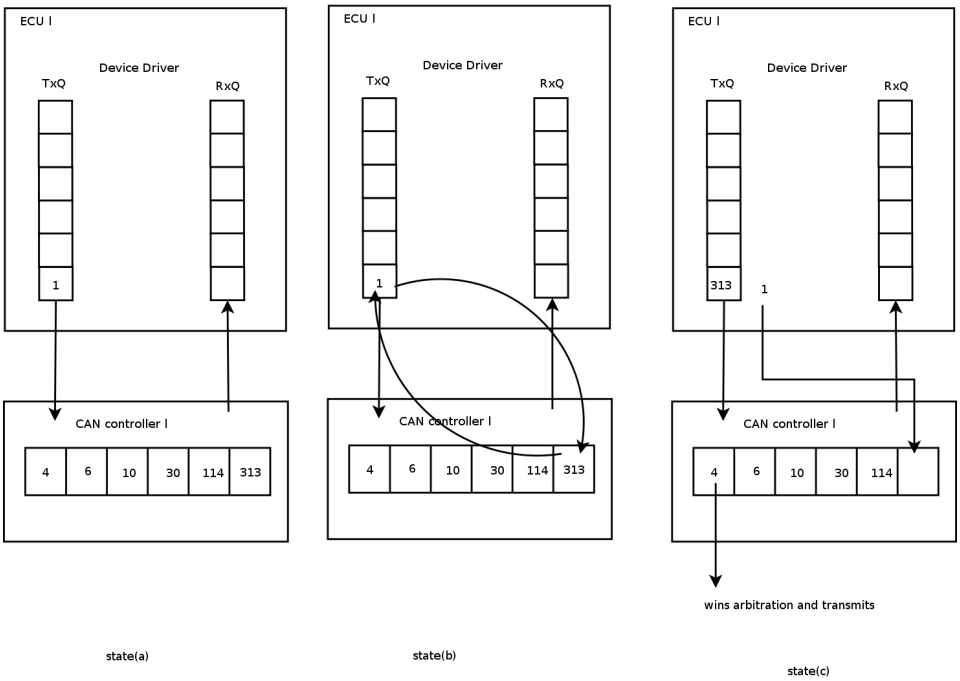


Figure 3.2: Priority inversion due to copy-time. In state(a) frame with *ID=1* gets released and since it has highest priority the driver decides to remove the lowest priority frame (*ID=313*) from the communication controller. In state(b) the driver starts to copy frame with *ID=1* in place of frame with *ID=313*. In state(c), while driver is copying frame *ID=1,* the arbitration starts and frame with *ID=4* wins the arbitration and begins to be transmitted. As frame *ID=1* has already been released, we have a priority inversion.

ones, the lowest priority message not under transmission in the transmission buffers is aborted and the new message is put in the transmission buffers. The message to be transmitted is stored in the transmit buffers. The CAN Driver confirms the transmit cancellation by the callback service and passes the old message back to the CAN Interface's priority queue, see figure 3.1 for details.

When any of these conditions does not hold, priority inversion occurs and the worst case timing analysis fails, meaning that the actual worst-case can be larger than what is predicted by existing analysis. However, a more subtle cause of priority inversion that may happen even when all the previous conditions are met. This problem arises because of the necessary finite copy time between the queue and the transmission buffers.

3.2.2 Implementation overhead(copy-time)

When all the transmission buffers in a CAN controller are filled and a message is released; assuming the newly released message is of lower priority than the messages in transmission buffer, then the newly released message waits in the priority queue

for the availability of one transmission buffer. However, if this newly released message is of higher priority than those in transmission buffers then - to respect the highest priority first (HPF) principle underlying CAN - it should be swapped with the lowest priority message in transmission buffers that is not undergoing transmission. Moreover, if the bus arbitration starts anytime during the swapping process (*i.e.*, lower priority message put back in the queue, higher priority message copied into the freed buffer), it may happen that a lower priority message, be it on the same station or elsewhere on the network, win the arbitration, as explained in figure 3.2, resulting in a priority inversion. The priority inversion suffered by the higher priority messages leads to the increase in the WCRT of those messages and this increase in WCRT is modeled by a factor called the Additional Delay (AD) in the rest of the chapter. An example of how AD occurs is shown in figure 3.1.

3.2.3 Single buffer with preemption.

Some CAN controllers have single transmit buffer, see table 3.1, which could be problematic.This case was discussed first in[Tindell 1994c]. Suppose on an ECU E_1 with single transmission buffer a message, μ_2, arrives at the queue right when message μ_3 started its transmission. The message μ_2 will have to wait for message μ_3 to complete its transmission before message μ_2 can be put in CAN controller transmission buffer for participation in an arbitration. This is unavoidable and considered as part of the blocking term B_1 . The copying of message μ_2 into transmission buffer will start when message μ_3 finishes its transmission.

However, if the message copy time message μ_2 is larger than the inter-frame bits (which can be further reduced because of clock skew on the CAN network), a new transmission of some lower priority message μ_4 on some other node can start while μ_2 is being copied. While μ_4 is transmitting, a new higher priority message $\mu_1$arrives on the same E_1 such that priority of $\mu_1 > \mu_2$ and the transmission request of μ_2 is thus aborted.

The message μ_1 can suffer same fate, described above, as that of message μ_2 and thus this priority inversion can happen multiple times, until the highest priority message from the ECU E_1, is written into the buffer and eventually transmitted.

3.2.4 Dual buffer with preemption

In [Meschi 1996] the discussion of the case of single buffer management with preemption was extended to the case of two buffers. Suppose on an ECU E_1 with two transmission buffers a message, μ_2, arrives and is put in a transmission buffer while message μ_3 started its transmission from other transmission buffer. Before the end of transmission for the message μ_3 another message μ_1 is released. Since the message μ_3 is under transmission and hence cannot be aborted, the message μ_2 will have to be aborted from its transmission buffer (since the priority of $\mu_1 > \mu_2$). However, during the time messages μ_2 and μ_1 are being swapped the transmission of message μ_3 can end and a lower priority message from some other node can win

arbitration, resulting in a priority inversion. This priority inversion scenario can repeat itself multiple times considering the fact that a new message of higher priority $\{\mu_k | k < 1\}$ can preempt message μ_2 right before message $\mu_1$ends its transmission, therefore multiple priority inversions.

It is argued in [Meschi 1996] that the only way to avoid having no buffer available at the time a new contention starts, which is ultimately the cause of priority inversion from lower priority messages, is to have at least three buffers available at the peripheral and sorted for transmission priority according to the priority of the messages contained in them. However, we will show in this work that such assumption may not necessarily be true.

3.2.5 FIFO message queue in a CAN driver

The limited number of transmission buffers inside a CAN controller was compensated by idea of using queues inside a CAN device driver to hold frames which did not find any available transmission buffer. However, these queues might follow FIFO queuing policy, for its simplicity, ease of implementation, easier queue management. However, when the queuing policy inside CAN driver is FIFO a higher priority message released will have to wait for the lower priority message at the head of the queue to copy itself first into emptied CAN transmission buffer. This is because with FIFO queues, preemption of the makes very little sense. In this case, a high priority message that is en-queued after lower priority messages will wait for the transmission of all the messages in front of it, see [Davis 2011a]. The delay suffered by a message in the queue will be directly proportional to the number of message in front of it in the queue, i.e. the messages en-queued before it. This can results in a priority inversion, and the substantial increase in the WCRT.

3.2.6 CAN controller message index

Ideally what we would have wanted for these CAN controllers was to transmit according to CAN ID. As can be seen in table 3.1 some CAN controllers may not provide most desirable behavior. These chips provide at least three transmission buffers (with an exception of Philips SJA1000) and the priority mechanism is independent from the CAN ID. This could lead to problem of priority inversion if the device drivers are not implemented in such a way to overcome this problem. For example in case of Micro-chip's MCP2515 assume the 2 buffers are filled with messages of priority 7 and 8, the CAN controller will assign the index of $(11)_b$ and $(10)_b$ respectively to these message. If a new message of priority 6 is released the indexes of messages have to be changed such that $6 := (11)_b, 7 := (10)_b, 8 := (01)_b$. The assignment of indexes is not automatic and has to be handled by the device driver, and if not take care of can result in a priority inversion. For example in case of above example if the message released in the end (message of priority 6) were assigned an index of $(01)_b$, it would have suffered priority inversion (as MCP2515 transmits highest index first). Some what similar issues exist in Freescale MC68HC912, but

unlike Micro-chip's MCP2515 it has an 8 bit index.

Moreover,these issue do not occur in Philips SJA1000 CAN controller as it has only one buffer, but it still retains the limitations of its predecessor, that is, a single output buffer and hence the susceptibility of priority inversion as discussed in subsection 3.2.3.

To overcome the issue of priority inversion because of CAN controllers own priority mechanism a proper care must be taken while implementing the device drivers to map the CAN ID to CAN controllers indexing and vice-versa, such messages get transmitted as lowest CAN ID first. Device drivers will also have to consult this map when placing or aborting a message in the CAN transmission buffers.

3.2.7 Impossibility to cancel message transmissions

In case the message cancellation is not possible, due to CAN controller not supporting it or device driving not supporting it, the higher priority messages released on an ECU may get blocked by the lower priority messages when all the buffers are filled resulting in a priority inversion [Khan 2011]. The priority inversion suffered by the higher priority messages leads to the increase in the WCRT of those messages and this increase in WCRT is modeled by a factor called the Additional Delay (AD) in the rest of the chapter. An example of how AD occurs is shown in figure 3.5.

This case arises when pending messages are sorted according to priority in a single queue. In addition, the transmission buffers cannot be aborted, that is, when a message is copied into it, the other messages in the queue need to wait for its transmission. The reason for non-abortion, as mentioned earlier, can be the driver does not support it or the CAN controller does not support it. In this case, the behavior is of the system becomes similar to that of a FIFO queue. As the messages in the priority queue may be blocked by a lower priority message waiting for transmission in the transmission buffers. This type of priority inversion clearly violates the rules on which were established in subsection 3.2.1.

3.3 System model

We assume a set $\mathcal{M}$ of m messages μ_1, μ_2, ..., μ_m, where $m \in \mathbb{N}$. Each message μ_i is characterized by a *period* $T_i \in \mathbb{R}^+$, an activation jitter $J_i \in \mathbb{R}^+$, a worst-case transmission time $C_i \in \mathbb{R}^+$, and a (*relative*) *deadline* $D_i \in \mathbb{R}^+$, where $D_i \leq T_i$. Moreover, one defines the maximum copying time CT_i for μ_i as the maximum between the time needed to copy the message from the queue to the transmission buffer and the time to copy from the buffer to the queue[3]. Here, we make the reasonable assumption that the copy-time is less than the transmission time of the smallest frame. Furthermore, we are assuming that multiple transmission buffers on CAN controllers are not occupied by messages of the same priority.

[3]Both delays could be distinguished but in practice we expect them to be very similar.

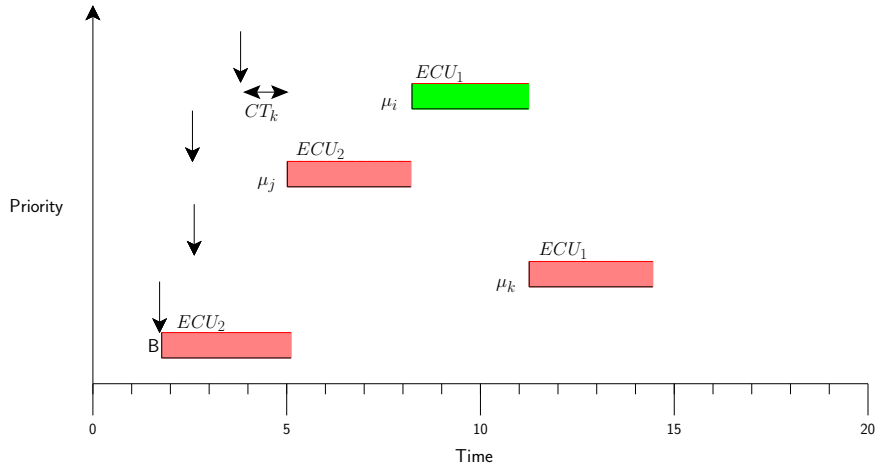


Figure 3.3: Message μ_i is released while a lower priority frame is being sent (blocking delay B). The transmission buffers on ECU1 are full, the device driver then aborts lower priority message μ_k and copies it into queue taking time CT_k. Then μ_i is copied into the freed transmission buffer taking time CT_i. However, while μ_i is being copied the arbitration is lost to message μ_j and μ_i suffers an additional delay of $AD = CT_k + C_j - B$ as compared to initial B. It should be pointed out that this additional delay of μ_i appears as an additional jitter to lower priority message μ_k.

For notational convenience, we assume that the messages are given in order of decreasing priority, i.e. μ_1 has highest priority and μ_m has the lowest priority. Moreover, we assume a set $\mathcal{C}$ of n CAN controllers $CC_1, CC_2, \ldots, CC_n$, where $n \in \mathbb{N}$. Each CAN controller CC_c has a finite number of transmission buffers $k_c \in \mathbb{N}$.

A total function $CC : \mathcal{M} \rightarrow \mathcal{C}$ defines which message is sent by which CAN controller. The set of messages M_c sent by controller CC_c is defined as

$$M_c = \{\mu \in \mathcal{M} | CC(\mu) = CC_c\}. \tag{3.1}$$

Similarly, $\overline{M_c}$ defines the set of messages *not* sent by CC_c, i.e.

$$\overline{M_c} = \{\mu \in \mathcal{M} | CC(\mu) \neq CC_c\} = \mathcal{M} \setminus M_c. \tag{3.2}$$

Let H_c be the set of highest priority messages in M_c excluding the k_c lowest priority messages. Similarly, let HE_c be the set of highest priority messages in M_c excluding the $k_c - 1$ lowest priority messages. We use μ_{L_c} to denote the lowest priority message in message set HE_c, where L_c is its priority. Furthermore, we assume that multiple transmission buffers on CAN controllers are not occupied by messages with the same priority. The assumption is made that nodes can always fill empty buffers with ready messages in time for the next arbitration.

The WCRT R_i of a message is defined as the maximum possible time taken by a message to reach the destination CAN controller, starting from the time of an initiating event responded to by the sending task. A message μ_i is said to be schedulable if and only if its WCRT R_i is less than or equal to the message relative

deadline D_i and the system is schedulable if and only if all of the messages are schedulable.

Priority inversion A message μ_i on a CAN controller CC_l without abort mechanism is said to suffer from priority inversion when μ_i is released, if all of the k_l transmission buffers are occupied by the messages with lower priority than that of μ_i.

Limited number of buffers For any CAN controller CC_l with k_l transmission buffers the k_l lowest priority messages in the message set M_l will not suffer any priority inversion. As a corollary, for any CAN controller CC_l with k_l transmission buffers, if the number of messages mapped onto it is less or equal to k_l then no message on CC_l can suffer from priority inversion.

3.4 Response time analysis: abortable case

This section provides the method to compute the worst-case response time of messages on the CAN network, when priority inversion due to copy-time is considered. The computed values are then used to check the schedulability of the system by comparing the WCRTs against the deadlines. The analysis given in this chapter provides a simple and non-necessary schedulability condition directly inspired from [Davis 2007]. It assumes no errors on the bus but they can be included as classically done in [Tindell 1995]. Following the analysis given in [Tindell 1995, Davis 2007] the worst-case response time can be described as a composition of three elements:

1. the queuing jitter J_i, the longest time it takes to queue the message starting from initiating event,
2. the queuing delay w_i, the longest time for which a message can remain in the driver queue or transmission buffers before successful transmission,
3. the worst-case transmission time C_i, the longest time a message can take to be transmitted.

A bound on the worst-case response time of a message μ_i is therefore given as:

$$R_i = J_i + w_i + C_i \tag{3.3}$$

The queuing delay w_i is composed as follows:

1. blocking delay which is the delay due a lower priority frame that has started to be transmitted before μ_i can participate to the arbitration, plus possibly the time needed to free a buffer on the ECU of μ_i (see section 3.4.2),
2. the delay due to interference of higher priority messages which may win the arbitration and transmit one or several times before μ_i.

When computing bound on the response times, we can distinguish two cases i) messages which are safe from priority inversion ii) messages which suffer from priority inversion and will be swapped with the lowest priority message in transmission buffers not in transmission.

3.4.1 Case 1: safe from any priority inversion

We note that the higher priority messages on each CAN controller CC_l are more susceptible to priority inversion than lower priority messages on the same CAN controller. Indeed, the k_l lowest priority messages on CC_l will not suffer from any priority inversion as not all of the transmission buffers can be occupied by messages with lower priority than any if these k_l messages, thus these messages are not suffering from any *additional delay*. However, these messages are still affected by the additional delay of higher priority messages, as it is seen by them as additional jitter. For these messages or the CAN controllers which support abort mechanisms, the worst-case queuing delay, using the model in [Davis 2007], is given by:

$$w_i^{n+1} = \max(B_i, C_i) + \sum_{\forall k < i \wedge \mu_k \in M} \left\lceil \frac{\hat{J}_k + w_i^n + \tau_{bit}}{T_k} \right\rceil C_k \tag{3.4}$$

where $\hat{J}_k$ is computed using (3.12) and B_i is the maximum blocking time due to lower priority messages which occurs when a lower priority message of the largest size has just started to be transmitted when μ_i arrives, i.e.

$$B_i = \max_{\forall k > i \wedge \mu_k \in M} \{C_k\} \tag{3.5}$$

A suitable starting value for the recurrence relation given above is $w_i^0 = C_i$. This relation keeps on iterating until $w_i^{n+1} = w_i^n$ or $J_i + w_i^{n+1} + C_i > D_i$, which is the case when the message is not schedulable. If the message is schedulable its WCRT is given by (3.14).

3.4.2 Case 2: messages undergoing priority inversion

Messages not belonging to the k_l lowest priority messages can suffer from priority inversions when all the k_l transmission buffers are filled up with lower priority messages. We consider here the case where the communication driver will abort a transmission request whenever a message that possesses a higher priority than those already in the transmission buffers arrives, let's say μ_i. Specifically, the CAN driver will abort the lowest priority message on CC_c *not currently under transmission* and start copying μ_i in place. The swapping of μ_i will induce some delay and if arbitration starts during the swapping process a lower priority message than μ_i may win arbitration and starts to transmit. This may introduce an additional delay AD_i for μ_i which is equivalent to the difference between the transmission time of the message which won arbitration and the original blocking delay B_i, plus the time needed to copy a message from the communication buffer to the queue. The worst-case AD_i is

obtained by taking the maximum of the worst-case transmission times for all values of k such that $i < k \leq j$ where μ_j is the highest priority message of the lowest k_l priority messages on CC_l:

$$AD_i = \max\left(0, \max_{\{\forall k \in M_C | k > i\}}(CT_k) + \max_{i<k\leq j}(C_k) - B_i\right) \tag{3.6}$$

where CT_k is the copy time of the message which is replaced by μ_i. Then, the worst-case queuing delay for message μ_i is given by:

$$w_i^{n+1} = \max(\hat{B}_i, C_i) + CT_i + \sum_{\forall j \in hp(\mu_i)} \lceil \frac{\hat{J}_j + w_i^n + \tau_{bit}}{T_j} \rceil C_j \tag{3.7}$$

where $\hat{J}_j$ is given by (3.12) and $\hat{B}_i$ is given by $B_i + AD_i$. A suitable starting value for the recurrence relation give above is $w_i^0 = C_i$. This relationship keeps on iterating until $w_i^{n+1} = w_i^n$ or $J_i + w_i^{n+1} + C_i > D_i$, which is the case when the message is not schedulable. And if the message is schedulable its WCRT will be given by (3.3).

3.5 Optimized implementation and case-study

If we accept the overhead of keeping a copy of the messages currently in the transmission buffers in the priority queue, we can suppress an extra copy time and remove the quantity $\max_{\{\forall k \in M_C | k > i\}} CT_k$ in (3.6). This can be done by maintaining an extra status field along with the priority queue. For instance, for the messages in the transmission buffers this field could be set to one and for the messages in priority queue but not in any transmission buffer this field could be set to zero. Upon the successful transmission of a message its corresponding copy along with its status field will be removed from the priority queue.

Upon a full transmission buffers, for any new message with priority greater than any message in the transmission buffers, it will be first put in the priority queue then the status field of message in transmission buffers with lowest priority and not transmitting will be set to zero. Then the message will over-write the message in transmission buffer whose field was just set to zero and finally for the message which replaced the message in the transmission buffer, the status field is set to one. This procedure will remove the need for swapping which takes more time as compared to simple overwrite and thus chances of priority inversion are reduced. However, the downside of this is that we have to re-arrange the priority queue not only each time a message becomes available but also each time a message is successfully sent by the station (upon the acknowledgment).

We illustrate the analysis on an typical 125Kbit/s automotive body network. To generate a realistic test network we used *Netcarbench* [Braun 2007]. The generated periodic message sets under study consists of 105 CAN messages mapped over 17 ECUs with deadlines equal to periods and data payload ranging from 1 to 8 bytes. The total periodic load is equal to 42.04%.

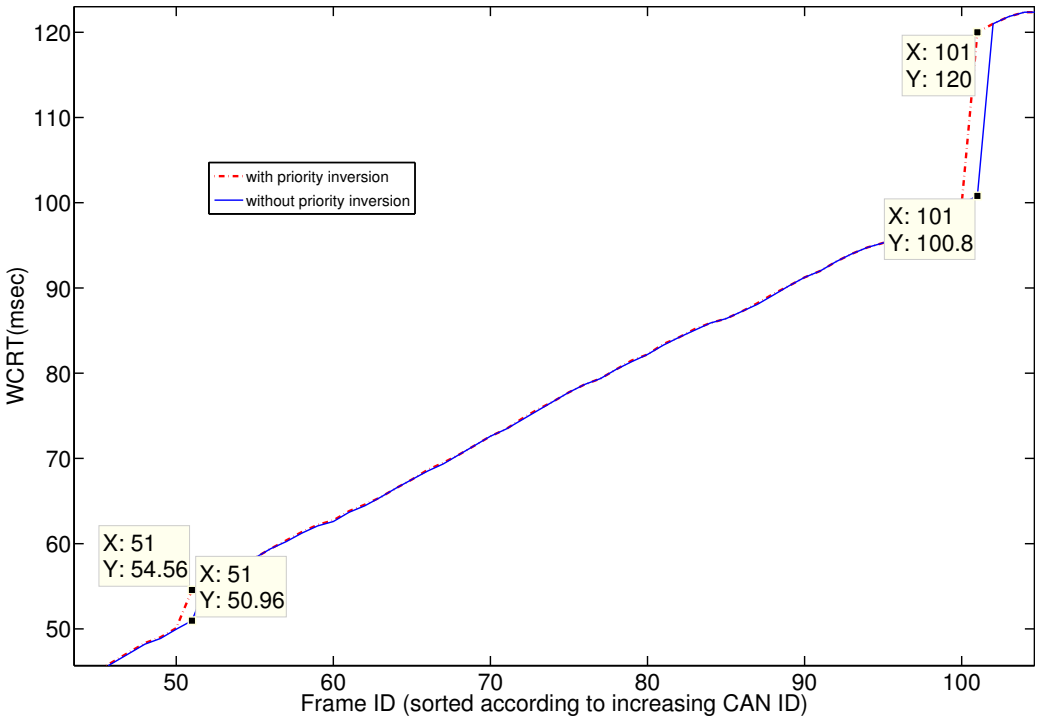


Figure 3.4: Worst-case response time with and without taking into account priority inversion. Only frames starting from ID 40 are shown.

Figure 3.4 shows the worst-case response times of the CAN messages with and without priority inversion. We observe the impact on the WCRT of messages when priority inversion is taken into account. For instance in figure 3.4, the WCRT for the message with id 101 raises from 100.8ms without priority inversion to 120ms (*i.e.* 19% increase).

3.6 Response time analysis: non-abortable case

This section provides the method to compute the worst-case response time of messages on the CAN network, when priority inversion due to non-abortion of messages is considered. The computed values are then used to check the schedulability of the system by comparing the WCRTs against the deadlines. The analysis given in this chapter provides a simple and non-necessary schedulability condition directly inspired from [Davis 2007]. It assumes no errors on the bus but they can be included as classically done in [Tindell 1995]. Following the analysis given in [Tindell 1995, Davis 2007] the worst-case response time can be described as a composition of three elements:

1. the queuing jitter J_i, is the maximum time between a task being released and a message being queued.

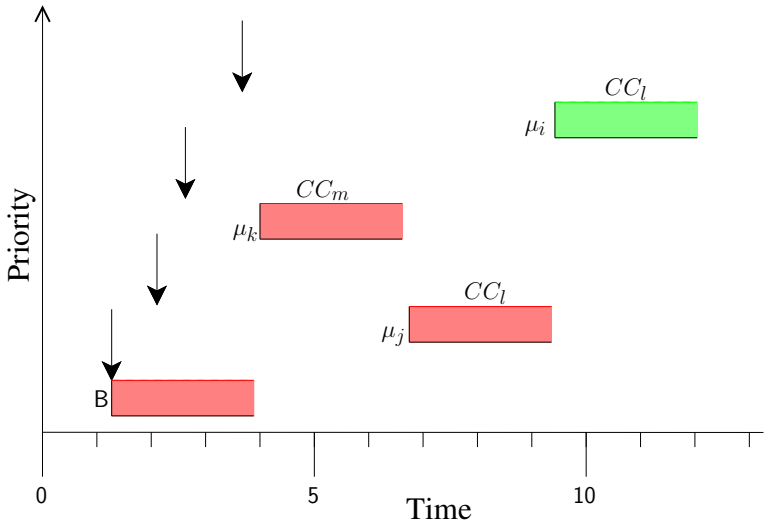


Figure 3.5: The message μ_i suffers a priority inversion as, being the highest priority message, it should have been transmitted earlier than μ_k and μ_j sent by nodes CC_m and CC_l respectively. This was not possible because here the transmission request for μ_j cannot be aborted on CC_l and all buffers were full. This results in an additional delay for message $\mu_i$and thus increased WCRT as compared to existing analyses. The arrows indicate the message release times.

2. the queuing delay w_i, the longest time for which a message can remain in the driver queue or transmission buffers before successful transmission,

3. the worst-case transmission time C_i, the longest time a message can take to be transmitted.

A bound on the worst-case response time of a message μ_i is therefore given by equation (3.3)

When computing bound on the response times, we can distinguish three cases i) messages which are safe from priority inversion ii) messages which suffer from priority inversion due to non-abortion of the messages in transmission buffers and iii) message which suffer from priority inversion due to copy-time and message swapping issue. We are analyzing second case here and the first and third case has been already analyzed in section 3.4.

3.6.1 Additional Delay

Figure 3.5 illustrates the case in which a message μ_i sent by CAN controller CC_l should have been transmitted after B, the blocking time of a lower priority frame. Here the message μ_j blocks μ_i due to the non-availability of a transmission buffer in CC_l, which only becomes available after μ_j finishes its transmission. However, the message μ_j has to wait for the higher priority message μ_k on CAN controller CC_m to be transmitted before it can begin its transmission. Therefore, the WCRT for μ_i

given by the existing analyses increases by an amount, called the Additional Delay (AD), which in this example is equivalent to the sum of the worst-case transmission times of μ_k and μ_j.

Let μ_i be a high priority message in M_c and let the number of messages in M_c with a lower priority than i be at least k_c. Moreover, let μ_j be the highest priority message in the CC_c transmission buffers, such that $j > i$ (i.e. j is of lower priority than i). When all the transmission buffers of CC_c are full, the longest delay for μ_i occurs when none of the messages in the transmission buffers of CC_c are currently being transmitted and μ_i has to wait until μ_j has been transmitted for the release of a buffer on CC_c. Moreover, μ_i also experiences the normal interference from higher priority messages sent by CAN controllers other than CC_c.

Algorithm 3 Algorithm for finding additional delay and additional jitter. The inputs to the algorithm are the number of CAN controllers (c), the number of transmission buffers on each CAN controller c (k_c), and the set of all messages on the CAN network (M). The algorithm returns the additional delay and additional jitter for all messages.

Input: c, $k = \{k_l | l = 1 \dots c\}$, M
Output: $AD = \{AD_i | i = 1 \dots size(M)\}, \hat{J} = \{\hat{J}_i | i = 1 \dots size(M)\}$
$AD = 0$ //initialization of AD for all messages
$\hat{J} = J$ //initialization of AJ for all messages
for *each* $CC_l | \, l \in \{1, 2 \dots, c\}$
  $K = size(M_l)$ //$size(M_l)$ returns # of messages in M_l
  $H_l = \{\forall \mu_i \in M | CC(\mu_i) == l \wedge i \leq K - k_l\}$ //set of messages with AD
  if $K \leq k_l$ //more buffers available than the # of messages
    $AD = 0$
  else
    $HE_l = \{\forall \mu_i \in M | CC(\mu_i) == l \wedge i \leq K - k_l + 1\}$ //message set H_l including μ_{L_l}
    compute $R_j^* \forall \, \mu_j \in HE_l$ //using equations (3.8 & 3.10)
    $\forall \, \mu_i \in < H_l$ *find* AD_i //using equation (3.11)
    $\forall \, \mu_i \in < H_l$ *find* $\hat{J}_i = J_i + AJ_i$ //using equations (3.12 & 3.13)
  end
end
return$(AD$ *and* $\hat{J})$

Before transmission (i.e. when μ_j is in the CAN controller transmission buffer blocking μ_i), μ_j can be directly blocked by at most one message μ_{l_j} with $l_j > j$ sent by another CAN controller, or alternatively, subject to indirect or push-through blocking due to at most one message μ_{l_j} with $l_j > j$ sent by the same CAN controller. Similarly, μ_j can experience interference from higher priority messages μ_{h_j} with $h_j < j$. Message μ_j cannot experience direct interference from higher priority messages μ_{h_j} with $h_j < j$ on controller CC_c, because μ_j is the highest priority message in the transmission buffers of CC_c and μ_j cannot be aborted. However, such messages

could if transmitted prior to the time at which μ_j fills the buffer, cause indirect interference by delaying the transmission of higher priority messages sent by other nodes, which then increases the time taken for message μ_j to be sent. To account for this indirect interference, we first include messages μ_{h_j} with $h_j < j$ on controller CC_c in the fixed point calculation of the queuing delay, so that the correct amount of interference is obtained for messages from other nodes. Later, when computing the additional jitter, we subtract out the interference from the messages sent by controller CC_c as these transmissions cannot occur after μ_j fills the transmission buffer.

Therefore the time duration for which μ_i has to wait depends on the response time of μ_j, called the modified response time[4] and denoted by R_j^* for μ_j and computed as follows

$$\hat{w}_j^{n+1} = \max(B_j, C_j) + \sum_{\forall \mu_k \in M \wedge k<j} \left\lceil \frac{\hat{J}_k + \hat{w}_j^n + \tau_{bit}}{T_k} \right\rceil C_k \tag{3.8}$$

where B_j is the maximum blocking time of message μ_j given by:

$$B_j = \max\{0, \max\{C_k | k > i\}\}. \tag{3.9}$$

Where $\hat{J}_k$ is the jitter[5] of higher priority messages computed using equation (3.12) by algorithm 3. A suitable starting value for the recurrence relation given in equation (3.8) is $\hat{w}_j^0 = \bar{B}_j$. This relationship keeps on iterating until $\hat{w}_j^{n+1} = \hat{w}_j^n$ or $\hat{w}_j^{n+1} + C_j > D_j$, which is the case when μ_j is not schedulable. The modified WCRT of μ_j is given by:

$$R_j^* = \hat{w}_j + C_j \tag{3.10}$$

There are some aspects that need to be taken into account in order to determine the additional delay experienced by μ_i, due to the non-availability of a transmission buffer. First, the jitter J_j of μ_j should not be accounted for in the modified WCRT R_j^* of μ_j, because that is irrelevant for the delay of μ_i as μ_j is already in the transmission buffer.

Second, because the interference of messages μ_{h_i} with $1 \leq h_i < i$ will re-appear when we compute the worst-case response time of μ_i, we have to *subtract* this interference from R_j^*, in order to prevent the double inclusion of interference from the messages μ_{h_i} with $1 \leq h_i < i$ sent by other CAN controllers (i.e. $\bar{M}_c$).

The *additional delay* AD_i of μ_i, due non-availability of transmission buffer, is therefore found by *subtracting* the interference of the messages μ_{h_i} with $1 \leq h_i < i$

[4] The modified response time of message μ_j is not its actual response time because the message jitter is missing.

[5] To begin with $\hat{J}_k = J_k$ for all messages, in order to find the first value of AD_i. After computing AD_i, it will appear as jitter to all messages $\{\mu_k | k > i\}$ necessitating recalculation of AD_i, which is done iteratively until it does not change any more or a message becomes unschedulable, found using algorithm 3.

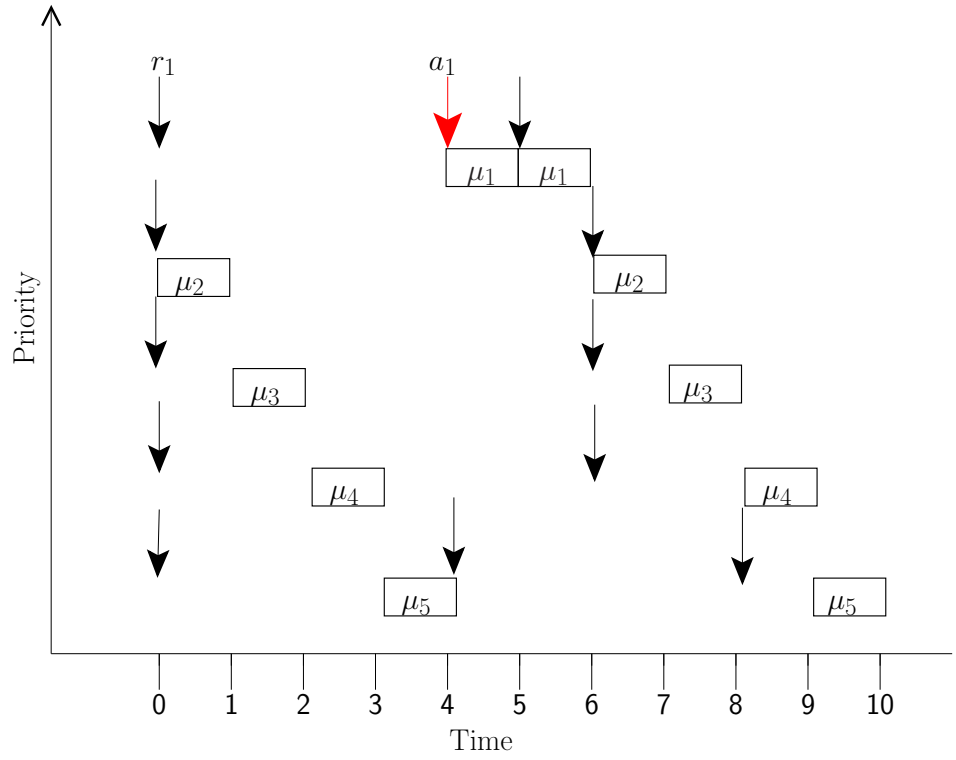


Figure 3.6: Example of how the WCRT of a lower priority message μ_5 is affected by the additional jitter caused by priority inversion that is suffered by a higher priority message μ_1.

and μ_{h_k} with $1 \leq h_k < j$ contained in R_j^*, i.e.

$$AD_i = \max_{\forall k > i \wedge \mu_k \in HE_c} \left(R_k^* - \sum_{1 \leq h_i < i \wedge \mu_{h_i} \in \bar{M}_c} \left\lceil \frac{R_k^* - C_k + \hat{J}_{h_i} + \tau_{\text{bit}}}{T_{h_i}} \right\rceil C_{h_i} - \sum_{1 \leq h_k < k \wedge \mu_{h_k} \in M_c} \left\lceil \frac{R_k^* - C_k + \hat{J}_{h_k} + \tau_{\text{bit}}}{T_{h_k}} \right\rceil C_{h_k} \right) \tag{3.11}$$

The reason for taking max in equation (3.11) is that the additional delay for the message μ_i can be due to each message $\mu_k \in HE_c$ where $i < k \leq L_c$, and it may be different due to each of these messages. Moreover, for all messages μ_k, such that $i < k \leq L_c$, having similar higher priority interference to that of μ_{L_c} (i.e. $R_k^* - C_k$ is equal to $R_{L_c}^* - C_{L_c}$) the worst-case AD_i is obtained by taking into account the message μ_k with the largest worst-case transmission time (*i.e.* $C_k > C_{L_c}$), as μ_k will give more additional delay than μ_{L_c}. Thus taking the maximum over all messages which could block μ_i enables us to find the message μ_k with $i < k \leq L_c$ which gives the worst-case additional delay to μ_i. The algorithm to find the additional delay is

Figure 3.7: The time line of message μ_i from its initiating event until it is able to participate in bus arbitration.

described in algorithm 3. The algorithm will keep on iterating until AD converges or it is greater than the deadline, *i.e.* WCRT of the message becomes greater than its deadline (in which case the message set is not schedulable).

3.6.2 Additional Jitter

The release jitter (J_i) is defined traditionally as the time interval between the occurrence of an event that will trigger sending of the message (r_i) and placing the message in a transmission queue (Q) or a transmission buffer. However, with non-abortable transmit buffers, priority inversion occurs, and the message μ_i triggered by the event at r_i is not able to participate in arbitration until the time a_i, as it may be blocked by messages with lower priority than i. Therefore, the messages on other nodes see the interference of μ_i after time a_i and the jitter of this message is not limited to J_i. Instead, the total jitter seen for μ_i, by the messages with lower priority than the priority of μ_i, is given by:

$$\hat{J}_i = J_i + AJ_i \tag{3.12}$$

where AJ_i is the time μ_i has to wait for the buffer to be emptied, see figure 3.7. Where AJ_i is computed as:

$$AJ_i = \max_{\forall k > i \wedge \mu_k \in HE_c} (R_k^* - \sum_{1 \leq h_k < k \wedge \mu_{h_k} \in M_c} \left\lceil \frac{R_k^* - C_k + \hat{J}_{h_k} + \tau_{\text{bit}}}{T_{h_k}} \right\rceil C_{h_k}) \tag{3.13}$$

where R_k^* is found using equation (3.10). Note that interference from higher priority messages sent by the same node is subtracted out, as this interference cannot occur after message μ_k has filled the transmit buffer. The above equation upper bounds the amount of time that a message μ_k can spend in a transmit buffer, with all other buffers filled by lower priority messages; hence it upper bounds the additional delay caused by message μ_k on message μ_i .

Example Consider a system of two CAN controllers CC_1 and CC_2 with 5 messages, as described in table 3.2. Let CC_1 have a single transmission buffer and let CC_2 have an unlimited number of transmission buffers. Assume that μ_5 is in the buffer of CC_1 and μ_1 is released along with all other messages at time $t = 0$, see figure 3.6.

Table 3.2: Characteristics of messages.

Frames	CAN controller	T	J	C
μ_1	CC_1	$5C$	0	C
μ_2	CC_2	$6C$	0	C
μ_3	CC_2	$6C$	0	C
μ_4	CC_2	$6C$	0	C
μ_5	CC_1	$4C$	0	C

Since CC_1 has a single buffer, μ_1 is blocked until μ_5 releases the buffer at time $t = 4$. The messages with lower priority than that of μ_1 on CC_2 are not aware of release at $t = 0$ of μ_1, as they do not see it participating in arbitration from $t = 0$ to a_1 when it occupies the buffer in CC_1. Once μ_1 is in the buffer it is able to participate in arbitration at time $t = 4$ and wins. The release of the second instance of message μ_5 suffers interference from two instances of message μ_1, between time $t = 4$ and $t = 6$. The inter-arrival time expected for μ_1 was $5C$, however, because μ_1 suffered an additional delay of $4C$ due to priority inversion, the interval between two instances of message μ_1 being sent on the bus is reduced to $1C$. The additional delay suffered by μ_1 is seen as a jitter of $4C$ by μ_5. The WCRT of μ_5 given by existing analyses is $5C$, but if we include the jitter of $4C$ for μ_1 we obtain the WCRT of $6C$ for μ_5 as seen in figure 3.6.

3.6.3 Response time analysis

This section provides a method for computing the worst-case response time of messages on the CAN network. The computed values are then used to check the schedulability of the system by comparing the WCRTs against the message deadlines. The analysis given in this section provides a simple and non-necessary schedulability condition directly inspired by [Davis 2007]. It assumes no errors on the bus but they can be included as done in [Tindell 1995]. Following the analyses given in [Tindell 1995, Davis 2007] the worst-case response time can be described as a composition of three elements:

1. the queuing jitter J_i, is the maximum time between the sending task being released and a message being queued.

2. the queuing delay w_i, is the longest time for which a message can remain in the device driver queue or transmission buffers before successful transmission,

3. the worst-case transmission time C_i, is the longest time a message can take to be transmitted.

A bound on the worst-case response time of a message μ_i is therefore given by:

$$R_i = J_i + w_i + C_i \tag{3.14}$$

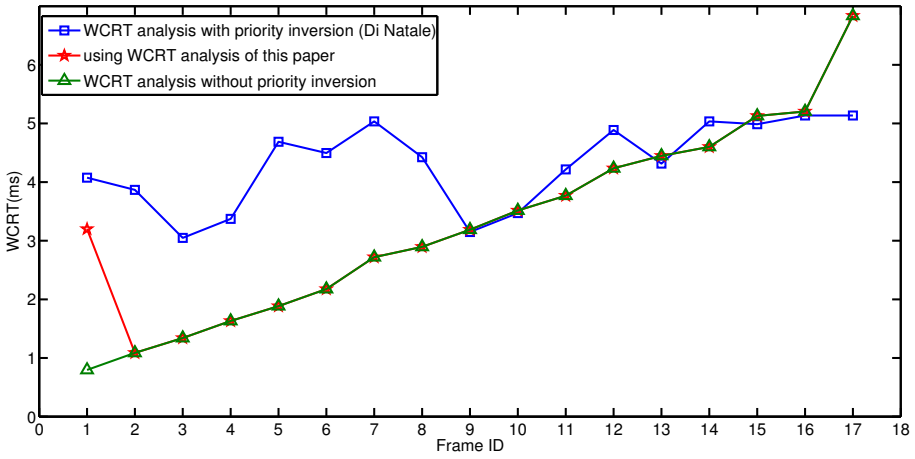


Figure 3.8: This figure shows the WCRT of messages from SAE benchmark computed using analysis which does not account for priority inversion, analysis in [Natale 2006] and the analysis developed in this section. Our analysis assumes each CAN controller has 3 transmission buffers. Some of the messages have lower WCRT with Di Natale's analysis (for example IDs 13, 15 and 17) because the equation used in [Natale 2006] to compute the WCET is slightly different.

The queuing delay $w_i$is composed of:

1. blocking delay[6] $\hat{B}_i$, is either the delay B_i due to the non-preemptivity of lower priority messages in transmission when μ_i was ready for arbitration or the additional delay AD_i, computed using equation (3.11), due to the priority inversion i.e.
$$\hat{B}_i = \max(\max(B_i, C_i), AD_i) \tag{3.15}$$

2. the delay due to interference of higher priority messages which may win arbitration and be transmitted before μ_i.

3.6.3.1 Case 3: not safe from priority inversion

Once we have the additional delay of message μ_i, susceptible to priority inversion, we can compute its WCRT. The worst-case queuing delay for message μ_i is given by:

[6]The additional delay AD_i of a message μ_i appears as an additional blocking delay due to messages with lower priority than that of μ_i.

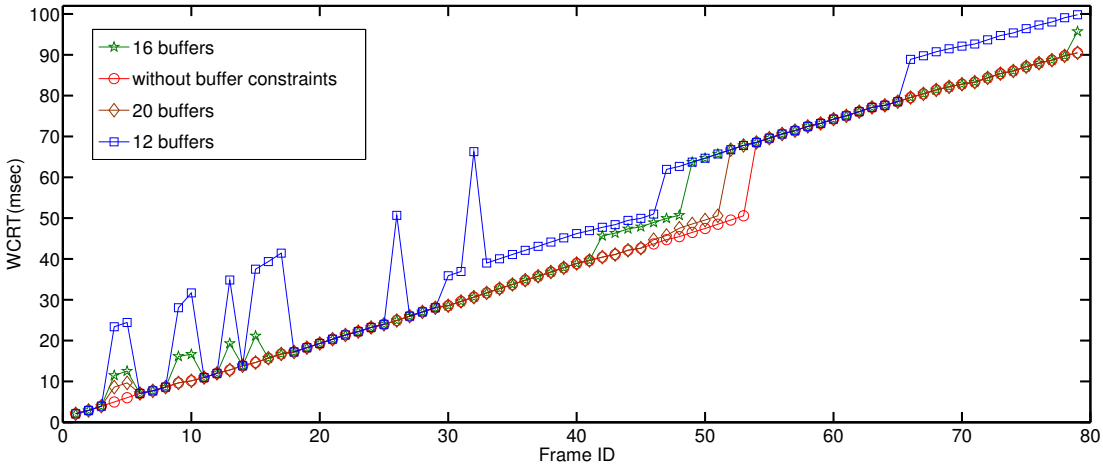


Figure 3.9: WCRT on a typical 125 kbits/s automotive body network (assuming each CAN controller has 12, 16 and 20 transmission buffers and cancellation of transmit request is not possible) computed using analysis which does not account for priority inversion (lower curve) and analysis developed in this section (section 3.6.3.1).

$$w_i^{n+1} = \hat{B}_i + \sum_{\forall k<i \wedge \mu_k \in M} \left\lceil \frac{\hat{J}_k + w_i^n + \tau_{bit}}{T_k} \right\rceil C_k \tag{3.16}$$

where $\hat{J}_k$ is computed using (3.12) and $\hat{B}_i$ is computed using (3.15). A suitable starting value for the recurrence relation given above is $w_i^0 = C_i + AD_i$. This relation keeps on iterating until $w_i^{n+1} = w_i^n$ or $J_i + w_i^{n+1} + C_i > D_i$, which is the case when the message is not schedulable. If the message is schedulable its WCRT is given by (3.14).

However, as we established in section 3.6.2 the computed additional jitter for μ_i now impacts all the messages with lower priority than i and therefore we have to re-compute the WCRT[7] for all lower priority messages as well.

The process used to re-compute WCRT for the messages remains the same as described in sections 3.4.1 and 3.6.3.1. A simple procedure is used to find the WCRT by computing additional delay first (for all messages susceptible to priority inversion) and then computing the WCRT for all of the messages, as shown in algorithm 4.

Example In section 3.6.2 we showed, with the aid of an example, how the additional delay of a message manifests itself as a jitter for lower priority messages and

[7]It is important to note that the additional delays effectively increase the jitter of affected messages, and this then leads to higher interference and a larger computed response time. However, in practice, the messages cannot obtain their maximum jitter (additional delays) all at the same time and therefore the analysis can be pessimistic. An improvement to the analysis is to upper bound the WCRT by the longest busy period at the lowest priority level, since no response time can be larger than that with any non-idling policy.

how existing analyses fail to integrate the same. We return to the same example to illustrate how the analysis developed in this section integrates the additional delay and the additional jitter. The message μ_1 is blocked by μ_5 and therefore the additional delay for μ_1 calculated using equation (3.11) is $4C$. The WCRT for μ_1 computed by equation (3.16) is $5C$. Similarly, the WCRT of message μ_5 when computed using equation (3.4) (by accounting for the additional jitter of message μ_1) is $6C$, which can be verified from figure 3.6.

We observe that the existing priority assignment algorithms, see [Davis 2011b], may not be optimal in this case as they require that the relative order among the higher priority messages does not matter while assigning priorities to lower priority messages. However, such a condition is not satisfied, for the scenario discussed in section 3.6.3.1, as the order among the higher priority messages may impact their additional delay, i.e. the jitter $\hat{J}$ seen by lower priority messages, thus have an impact on the response time of lower priority messages.

Algorithm 4 Algorithm for finding WCRT. The inputs to the algorithm are the number of CAN controllers (c), the number of transmission buffers on each CAN controller c (i.e. k_c), and the set of all messages on the CAN network (M). The algorithm returns the WCRT of message set.

Input: c, $k = \{k_l | l = 1 \dots c\}$, M
Output: WCRT of message set M
AD, $AD^{old} = 0$ // initialization of AD for all messages
$AD^{new} = C$
$\hat{J} = J$ // initialization of jitter for all messages
while(AD^{new}not equal to AD^{old})
    $AD^{old} = AD^{new}$
    Compute $\hat{J}$, AD^{new} via algorithm 3
    if(AD^{new} is greater than deadlines)
        return(unschedulable)
    end
end
$AD = AD^{new}$
if$(J + w^{n+1} + C \leq D)$ //for case 1 and case 2 using equations (3.14, 3.4 & 3.16)
    return$(J + w^{n+1} + C)$
else
    return(unschedulable)
end

3.7 Comparative Evaluation

The analysis developed in section 3.6.3.1 is compared against the existing analyses which do not account for priority inversion, and the analysis developed

in [Natale 2006] which accounts for priority inversion. The case-study assumes 3 or more transmission buffers on each CAN controller, with non-abortable transmission requests.

3.7.1 SAE benchmark

The evaluation of the analysis developed in section 3.6.3.1 is done by comparing against SAE benchmark results published in [Natale 2006] and in [Tindell 1994b]. The SAE benchmark, see [Tindell 1994b, Natale 2006] for details, describes a message set mapped on to seven different CAN controllers in a prototype car and the requirements for the schedulability of the messages. The network connecting the car subsystems handles 53 periodic and sporadic real-time signals. The signals have been grouped and the entire set has been reduced to 17 messages (for details, refer to [Tindell 1994b]). To analyse the schedulability of the message set at $250\,kbps$ we compute the worst-case transmission time for this bus-speed, which for consistency is computed as in [Natale 2006]. The results of the comparative WCRT analyses have been depicted in figure 3.8. The message set is schedulable with the analysis given in [Natale 2006] and with the analysis provided in section 3.6.3.1. However, a significant difference in the response time computed by the analysis in section 3.6.3.1 and the analysis in [Natale 2006] can be observed in figure 3.8. The reason for such a difference is that the analysis in [Natale 2006] does not consider the number of transmission buffers and computes the additional delay of the messages using the lowest priority message from the message set mapped onto that CAN controller, thus resulting in a pessimistic WCRT. Moreover, it has been established in [Khan 2010] and [Khan 2011] that the number of transmission buffers does have an effect on the WCRT. Applying the criteria developed for priority inversion in section 3.6.3.1 we find only one message in the benchmark may suffer from priority inversion ($ID = 1$), since there is only one CAN controller that has more than three messages mapped to it (see message mapping details in [Natale 2006]). Thus, the WCRT only increases for the message with $ID = 1$ as the rest of the messages are safe from priority inversion and they only take into account the additional jitter of the message with $ID = 1$. The worst-case of message $ID = 1$ is when the transmission buffers are filled with messages of $ID = 8, 12, 15$. The first message to transmit from the buffers is then $ID = 8$, which contributes towards the worst-case additional delay for message $ID = 1$, as in the worst-case it may have to wait for higher priority messages from other CAN controllers to be transmitted first (i.e. $ID = 2, 3, 4, 5, 6, 7$ contribute additional delay, computed using equation (3.11)).

3.7.2 Automotive body network

The limitation of the SAE benchmark is that it is outdated with respect to current in-vehicle systems. Moreover, the SAE benchmark has only one node with more than 3 messages mapped onto it, thus making it difficult to compare the analyses. Therefore, we illustrate the new analysis on an typical 125Kbit/s auto-

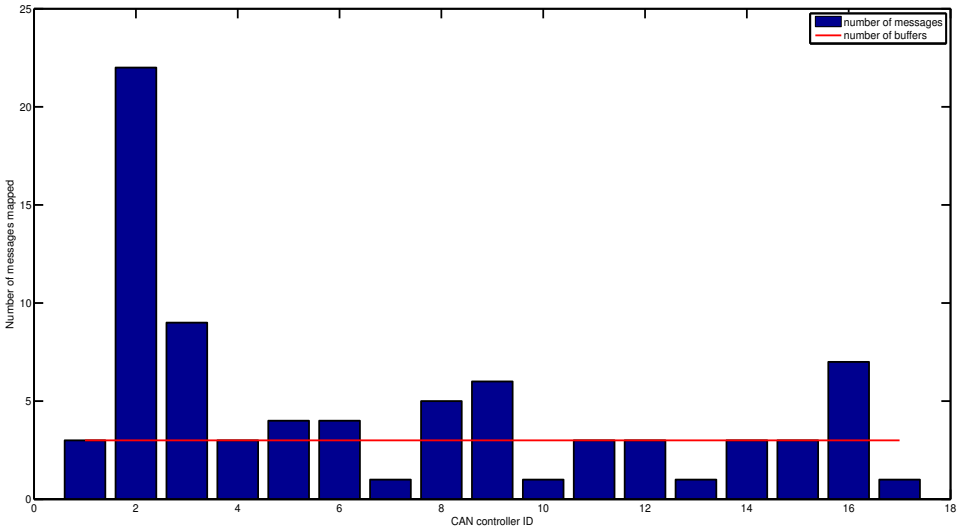


Figure 3.10: Figure showing number of messages mapped onto each CAN controller. The CAN controllers with more messages than the number of transmission buffers are susceptible to priority inversion.

motive body network. To generate a realistic test configuration we used the *Netcarbench* [Braun 2007] benchmark generator. The generated periodic message set under study consists of 79 CAN messages mapped over 17 ECUs with deadlines equal to periods and data payload ranging from 1 to 8 bytes. The total periodic load is equal to 64.26%. Figure 3.10 shows the message load distribution over the ECUs highlighting the ECUs with more than three messages susceptible to priority inversion, in the case where each node has three buffers. Figure 3.9 shows the worst-case response time of the CAN messages with and without priority inversion. We observe the impact on the WCRT of messages when priority inversion is taken into account. For instance, the message set is unschedulable when 3 transmission buffers per node is considered. Moreover, in figure 3.9, the WCRT for the message with ID=32 when considering 12 transmission buffers raises from 30.64ms without priority inversion to 66.29ms. The underlying reason for such an increase in the WCRT is the additional delay of 19.46ms encountered by frame ID=32. This is because the frame which is blocking message ID=32 in the worst-case scenario has ID=69 and the number of frames on other ECUs having ID between ID=69 and ID=32 is 27. Therefore, in the worst-case additional delay scenario, 27 messages may be transmitted before message ID=69 could be transmitted and then subsequently release the buffer for message ID=32.

We also note that the choice of priorities greatly influences the amount of additional delay. For example, if the priorities were such that the message blocking the message with ID=32 in worst-case had ID=44, then the number of messages on other ECUs blocking message ID=32 would have been reduced to 10 from 27, resulting in a smaller additional delay.

3.8 Summary

The aim of the chapter is to understand and analyze the consequences of architectural limitations in CAN. The chapter provides a model of schedulability analysis for CAN controllers when finite copy-time of messages is considered and when the transmission buffers can not be aborted. The model developed in this chapter provides very important understanding of the consequences due architectural limitations in CAN. Here, we derive a more realistic response time analysis in the typical case where controllers have three or more transmission buffers and the ability to cancel transmission requests is absent. This analysis is of particular interest to automotive sector where multiple Tier 1 suppliers provide ready to use ECUs in an automobile. And the lack of knowledge at system design level about the limitations of CAN controller used or device driver provided by tier 1 suppliers can have serious consequences. A first follow-up to this work is to come up with an analysis valid in the arbitrary deadline case. Another direct follow-up to this study is to investigate the case where, due to a larger message copy time, the nodes are not always able to fill empty buffers with ready messages in time for the next arbitration. As seen in case study of section 3.4 the implementation quality and the architecture of the CAN device driver can have consequences on the WCRT of messages and we provide the some guidelines to avoid the same. Also, as seen in the case-study of section 3.6 the choice of priorities has an effect such that the additional delay gets reduced,therefore as a future work we will study the priority mapping schemes which could reduce the amount of additional delay in case a message suffers from priority inversion. Also, we will study the choice of offsets on ECUs so that messages are not released at the very same moment, to reduce the chances of priority inversion in a CAN controller.

Chapter 4

Probabilistic Analysis for Component-Based Embedded Systems

Contents

In this chapter we present a novel analysis for complex real-time systems involving component-based design and abstraction models. The abstraction that we develop allows us to analyze the system having mixed components (i.e., both deterministic and probabilistic components). The deterministic and probabilistic models of the components are abstracted through the interfaces based on the curves having *probabilistic bounds* associated with them. The resulting component framework allows us to analyze the mixed (probabilistic and deterministic) component system. The *probabilistic bound* of the interface (abstracted by curves) allows us to differentiate between real-time guarantees (such as hard and soft) in the analysis (based on the safety requirements and system specifications). In the end we present a test case to show how the proposed analysis framework can be used to address the different safety requirements while modeling the real-time systems.

4.1 Introduction

The ECUs in an automotive systems are embedded and interacting with the physical system, forming the a system of complex nature. Moreover, with the proliferation of ECUs in automobiles, the complexity of the automotive embedded systems (AESs) has risen to the level never considered before, mostly because of the complex nature of operational environment and the large number of elements, exploiting functional and non-functional aspects, which compose the systems. The complexity of AESs necessitates the advanced design and analysis methods to assure temporal requirements. The complexity is, therefore, a key reason for finding the alternative and efficient abstractions of AESs. The abstraction frameworks have been applied with the purpose of analyzing complex real-time systems (such as AESs) and their timing requirements [Chakraborty 2003, Shin 2003, Mok 2001]. Besides the abstractions, component-based design has been widely accepted as an approach to facilitate the design of complex real-time systems [Lorente 2006, Shin 2004b]. It provides means for decomposing a complex system into simpler components, thus simpler design problems. The components are then composed into a system using interfaces. The composition through the interface guarantees that the analysis performed at the component level holds for the system as well, i.e. when a system is composable. Simply put, the component interfaces abstract the component-level timing requirements and allow to check compliance to non-functional constraints of systems at composition time. However, such abstractions work for deterministic systems or the systems where we have all the modeling parameters (such execution time, periods etc) available, in order to be able to analyze the system. Which is not necessarily true at the beginning of the automotive developmental life cycle. Since, all we may have at the early stage of development is the timing budget provided by the OEMs formed by decomposing the end-to-end latency. Therefore, we need an analysis framework which can handle complexity, in terms of lack of modeling data, such that it allows the designer to do better dimensioning of the systems.

The basic rationale for performing the probabilistic analysis of real systems is that it is difficult to provide hard real time guarantees, since the neither the behavior of the design nor the hardware components can be completely guaranteed [Hansson 2002]. Nevertheless, the timing analysis of such systems has been extensively studied by considering worst-case values that induce a certain pessimism, like over dimensioning of the system, which cannot be afforded in automotive domain. Another rationale to be considered is that the hardware and software elements composing RTSs may usually experience or exhibit some randomness. For example failures due to Electro Magnetic Interference (EMI), aging of hardware components, probabilistic execution times, and choices in randomized algorithms. Due to these reasons, establishing the temporal correctness, the composability and the scalability of these systems under all circumstances is usually expensive, thus impractical. For these cases other approaches could be taken into account such as the probabilistic approaches. Moreover, the unreliable nature of the system environment and the system elements may pose a serious problem in safety critical applications, such as

those in applications for space, military, automotive and medicine. The performing probabilistic analysis become more useful as the quantification of these measures (safety, reliability) given by various standard is done through probabilistic threshold values. Thus developing a component probabilistic analysis framework serves the purpose of reducing system complexity and being able to perform better dimensioning of the system, when not a lot of modeling data is available. Such an approach is very interesting, as well, as it ensures refined results as we refine the modeling data (as and when it become available), without having to make any changes to the analysis framework.

4.1.1 Deterministic component models

A component-based view of real-time systems is defined such that each system element can be modeled as a component [Chakraborty 2003, Shin 2004b, Easwaran 2006, Lorente 2006]. The component interface describes how the component relates to other components as well as the environment in terms of inputs/outputs [de Alfaro 2001, de Alfaro 2005]. In particular, real-time interfaces codes the timing requirements of the component [Shin 2008a, Wandeler 2005]. There are various techniques which have been developed. However, here we are interested in the real-time calculus (RTC) [Thiele 2000], derived from network calculus [Le Boudec 2001]. Which is a worst-case analysis framework for real-time systems based on deterministic bounds. The bounds model the system timing behavior. The RTC allows event occurrences to be related to the passage of quantitative deterministic time: non-deterministic decisions can be taken throughout bounding curves. The RTC supports component-based design and analysis of real-time systems; where the schedulability analysis is carried out at design time through real-time interfaces [Thiele 2006, Wandeler 2006a]. Where as the Component design paradigm [Shin 2004a, Shin 2008b] provides the mechanism to compose large and complex real-time systems from independent sub-systems.

4.1.2 Probabilistic analysis of real-time systems

The probabilistic approach [Burns 2003] allows probabilistic choices to be defined, rather than the simple deterministic/non-deterministic choices. Consequently, there is the need to extend abstractions and classical analysis methods in terms of probabilistic parameters and bounds, i.e., a resource curve and a probability associated representing a *bound to the resource provided* and *the probability that the curve bounds the resource actually provided*, respectively. The probabilistic analysis does not introduce any worst-case or restrictive assumptions into the real-time analysis and its applicable to general priority-driven systems. The probabilistic models of real-time systems consider the systems to have at least one parameter described by a random variable. Among the studies in this area, we mention [Navet 2000, Navet 1998, López 2008, Zeng 2009, Díaz 2002, Cucu 2006], which tackle with different random parameters of real-time systems.

In this chapter we apply the probabilistic model to abstract the curves, which defines the interface of a components. Where the curves represent the cumulative amount of work to be performed or cumulative processing power available. The abstraction of curves have been performed made with the stochastic network calculus [Jiang 2006]. However, the stochastic network calculus does not provide information for real-time analysis or any guarantees as such. Moreover, In [Santinelli 2011] authors have developed a probabilistic extension to the real-time calculus [Thiele 2000], for performing schedulability analysis of real-time systems (considering the execution time and period to be random). The work in [Santinelli 2011] was done in parallel with the work presented in this chapter. In comparison to [Santinelli 2011], this work develops the theory for task arrival characterization based on probabilistic model of aperiodic arrivals. Moreover, this work introduces the concept of a probabilistic interface and then developing compositional framework thereafter. In [Santinelli 2011], the probabilistic bounds are modeled as functions and requires convolution operation to find the residual probabilities. In comparison this work models the probabilistic bounds as simple values which are easy to compute using simple arithmetic operations. Besides, in this work we show how to find underlying distribution of a process and then how to get curves from that. This work also differs by the introduction/integration of safety levels into the compositional framework developed.

4.1.3 Safety critical systems

The proliferation of critical embedded systems has an impact on the safety, as these systems inherit the safety properties of the mechanical system being replaced (for example, brake-by-wire). Moreover, such a proliferation has resulted in the increased sophistication, heterogeneity and complexity in the networks, besides increasing the levels of subsystem integration. Therefore, there is a growing need to ensure that AESs have reliability, availability and safety guarantees during normal operation or at critical instances (e.g. airbags during collision), despite of being in harsh environment with heat, humidity, vibration, electro-static discharge (ESD) and electro-magnetic interference (EMI). There are several well-established standards that provide guidelines and requirements for safety-critical systems. Among these standards, standards such as $IEC61508$ (industrial systems), $DO-178B$ (aircrafts) and $EN50128/9$ (railway transportation systems), assign a criticality level to a certain function/system based on the severity of a failure. The level of safety required depends on the criticality of the function to be performed by the system/function or a certain reliability expected from the system, expressed as a maximum probability of critical failure per hour. This safety level must be guaranteed in-order to be classified as the system of that guaranteed safety-level. We will illustrate how these reliability levels can be handled and verified with the framework developed in this chapter and this will be illustrated with the Safety Integrity Levels (SIL), defined in $IEC61508$. In this chapter, we use SIL which assigns the probability of failure on demand to each level of criticality; this probability is used as a threshold

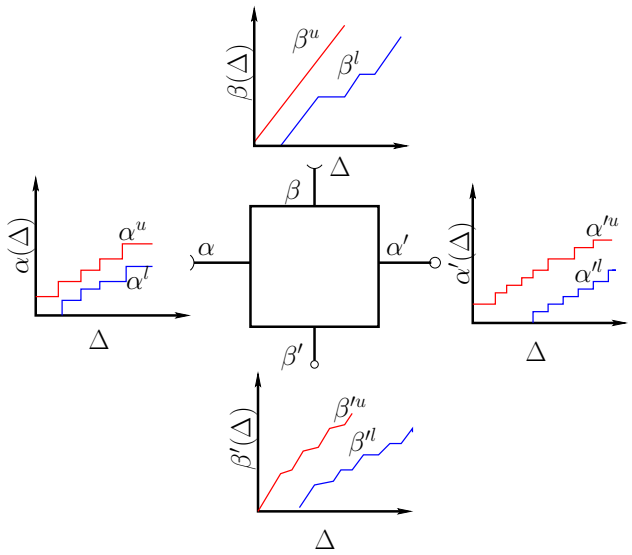


Figure 4.1: Example of a component with input curves such that the amount of work to do represented is by α and the amount of service available is represented by β. Similarly, for the output curves the remaining service is represented by β' and the output workload for subsequent component is represented by α' .

probability and will be simply called probability bound in the rest of the chapter[1].

Contribution of the chapter. In this chapter we develop a component based probabilistic analysis framework for analyzing complex AESs. The framework is based on the development of a probabilistic real-time calculus. The approach is based on the probabilistic bounds on the resource provisioning and resource demands for a generic real-time system. We then define a probabilistic component, in terms of its probabilistic interface, showing the conditions that are necessary for the composition of probabilistic components (composability). We then introduce the notion of safety with the probabilistic bounds. This provides a mechanism to include the safety standards into the developed analysis, in order to provide guarantees on the timing constraints of each real-time component and consequently the whole real-time system, in an safety critical paradigm. Finally, we also give the schedulability conditions for the probabilistic RTS. We also demonstrate the usefulness of our framework by co-analyzing a system with both probabilistic and deterministic properties, which may be true in large diverse systems.

4.2 Component model

The component-based view of real-time systems models each system element as a component [Lorente 2006, Shin 2003], and the component interface describes how

[1]This approach remains valid for other safety critical standards as well, and hence can be used with them.

the component relates to the other components and the environment in terms of functional and non-functional aspects. The behavior of a component can be modeled in terms of *arrival* and *service* curves which respectively abstract the resource demand and the resource provisioning for that component in the interval domain [Thiele 2000]. The figure 4.1 shows a generic component with the input and output curves (on an interface of the component). The component may not necessarily have an output workload curves (i.e. no interface on that side), i.e. α', when the component does not generate any resulting events against the input events, for example in a component abstracting a task which consumes the event but does not generate an output event on a processing element. However, we can have output workload curves, for example, in case of component abstracting a communication resource which process an input event and then transmits an output event for the subsequent component. We also assume that the output events abstracted by residual arrival curves α' are of same size (unit size) as that of input events abstracted input arrival curves, which can be easily generalized to arbitrary choices as is done in [Chakraborty 2003]. The relationship between α, β, α' and β' depends on the internal semantics of the component. For a generalized embedded system we assume that a component abstracts a task which is activated by an event and greedily consume the resource [Chakraborty 2003, Chokshi 2008]. We assume that the internal semantics of the component does not introduce any random behavior.

The concept of arrival and service functions come for network calculus and can be formalized as [Le Boudec 2001]:

Consider a function $f : \mathbb{R} \rightarrow \mathbb{R}^+ \bigcup \{+\infty\}$ such that $f(t)$ represents the amount of cumulative workload or service (available or requested) at given point of a component in the time interval $[0, t)$. The system is considered to be empty at $t = 0$. Therefore, $f(t)$ is a non-decreasing function of t with $f(t) = 0$ for $t < 0$.

Definition We define $\mathbb{F}$ as the set of all cumulative non-decreasing functions such that $\mathbb{F} = \{f : f(t_1) \geq f(t_2), \texttt{if } t_1 \geq t_2, \texttt{and } f(t) = 0, \forall t < 0\}$

Therefore, if R and C represent cumulative arrivals and cumulative service functions respectively then $R, C \in \mathbb{F}$.

4.2.1 Workload model

We model aperiodic events with a stochastic process which counts the number of aperiodic events arrivals in a time interval. Let $\mathbb{X}$ be the cumulative distribution function (CDF) of the stochastic process which counts/gives the number of arrivals in the time interval $[0, t)$. Following definitions follow from the work presented in chapter 2. Where we modeled the aperiodic traffic as arrival curves. However, here we extend the definitions to introduce two classes of the curves. Which are upper and lower binding the aperiodic arrivals.

Definition [Upper cumulative arrival function] The "largest " cumulative function $R^+ \in \mathbb{F}$ such that $R(t)^+ = \sup\{R(t) | P[\mathbb{X}(t) \geq R(t)] \leq \Omega\}$.

Where Ω is a probability bound guaranteeing that CDF $\mathbb{X}$ gives higher cumulative arrivals with a probability of Ω.

Definition [Upper arrival curve] Given a non-decreasing non-negative request curve α^u we say that R^+ is constrained by α^u if and only if for all $s \leq t$: $R^+(t) - R^+(s) \leq \alpha^u(t-s)$.

Therefore, we can say R^+ has α^u as an arrival curve.

Definition [Lower cumulative arrival function] The "smallest " cumulative function $R^- \in \mathbb{F}$ such that $R(t)^- = \inf\{R(t)|P[\mathbb{X}(t) < R(t)] \leq \Omega\}$ [2].

Definition [Lower arrival curve] Given a non-decreasing non-negative request curve α^l we say that R^- is constrained by α^l if and only if for all $s \leq t$: $R^-(t) - R^-(s) \leq \alpha^l(t-s)$.

Therefore, we can say R^- has α^l as an arrival curve. The tuple $\alpha(\Delta) = [\alpha^u(\Delta), \alpha^l(\Delta)]$ of upper and lower arrival curves provides an arrival curve model, representing all possible curves of an event stream, where Δ is a time interval. Thus, for a time interval Δ we are guaranteeing the maximum arrivals of α^u and the minimum arrivals of α^l.

The probabilistic arrival curve at an interface of a component is represented by the couple $\langle$ *curve*, *probability bound* $\rangle$, such as $\gamma = \langle \alpha, \Omega \rangle$, as the curve α and its probabilistic bound Ω. The probability value $\Omega = 0$ for a curve represents the deterministic case or true bound. The process of finding the underlying distribution and finding the probabilistically bound function, such as $R(t)$, has been explained earlier in chapter 2 (same is true for $C(t)$ in resource model).

4.2.2 Resource model

The probabilistic service (resource) is modeled by a stochastic process having CDF $\mathbb{Y}$, which gives the amount of service available in the time interval $[0, t)$.

Definition [Upper cumulative resource function] The "largest " cumulative function $C^+ \in \mathbb{F}$ such that $C(t)^+ = \sup\{C(t)|P[\mathbb{Y}(t) < C(t)] \leq \Lambda\}$.

Where Λ is a probability bound guaranteeing that CDF $\mathbb{Y}$ gives lower cumulative arrivals with a probability of Λ.

Definition [Upper resource curve] Given a non-decreasing non-negative resource curve β^u we say that C^+ is constrained by β^u if and only if for all $s \leq t$: $C^+(t) - C^+(s) \leq \beta^u(t-s)$.

Therefore, we can say C^+ has β^u as an resource curve.

[2] where $R^-(t)$ is found from Complementary Cumulative Distribution Function (CCDF), where CCDF is defined as: $X_c(t) = P[X(t) < R(t)] = 1 - X(t)$.

Definition [Lower cumulative resource function] The "smallest " cumulative function $C^- \in \mathbb{F}$ such that $C(t)^- = \inf\{C(t)|P[Y(t) \geq C(t)] \leq \Lambda\}$.

Definition [Lower resource curve] Given a non-decreasing non-negative request curve β^l we say that C^- is constrained by β^l if and only if for all $s \leq t$: $C^-(t) - C^-(s) \leq \beta^l(t-s)$.

Therefore, we can say C^- has β^l as an arrival curve. The tuple $\beta(\Delta) = [\beta^u(\Delta), \beta^l(\Delta)]$ of upper and lower resource curves provides an resource curve model, representing all possible resource curves, where Δ is a time interval. Thus, for a time interval Δ we are guaranteeing the maximum resource of β^u and the minimum resource of β^l. The probabilistic service curve is represented by the couple *curve* and *probabilistic bound* as, $\eta = \langle \beta, \Lambda \rangle$. The probability value $\Lambda = 0$ for a curve represents the deterministic case or true bound.

4.2.3 Residual workload and resources

The figure 4.1 shows a component whose input interface is defined by the curves α and β, entering the component. The component processes workload α using the available resource β. The components generates the outputs, after processing inputs, on the output interfaces of the component. The resulting output curves are described by α' and β' (also called residual curves), The residual service β' is the remaining service, i.e. service remaining from β after serving the component. While as the residual arrival curve α' may not be necessarily present in a component, for example in a component which does not generate any output events against the input events. However, if a component is abstracting a task which greedily consumes the resource and generates output events against the input arrivals, we will abstract the residual arrival curves of such a component with α' [Chakraborty 2003].

Therefore, given the probabilistic arrival curves and resource processing this request, we can find then residual arrival curve $\langle \alpha', \Omega' \rangle$ and residual resource curve $\langle \beta', \Lambda' \rangle$ of the processing component as [Chakraborty 2003]:

$$\alpha'^l(\Delta) = \min\{\inf_{0 \leq u \leq \Delta}\{\sup_{v>0}\{\alpha^l(u+v) - \beta^u(v)\} + \beta^l(\Delta - u), \beta^l(\Delta)\}\} \quad (4.1)$$

$$\alpha'^u(\Delta) = \min\{\sup_{v>0}\{\inf_{0 \leq u \leq \Delta+v}\{\alpha^u(u) + \beta^u(v+\Delta-u)\} - \beta^l(v), \beta^u(\Delta)\}\}. \quad (4.2)$$

$$\beta'^l(\Delta) = \sup_{0 \leq v \leq \Delta}\{\beta^l(v) - \alpha^u(v)\} \quad (4.3)$$

$$\beta'^u(\Delta) = \min\{\inf_{v>0}\{\beta^u(v) - \alpha^l(v)\}, 0\}. \quad (4.4)$$

The bound on the residual *curves* is obtained through min-plus algebra [Le Boudec 2001]. These results are based on generalizing ideas from network calculus and hold specifically for infinite event streams [Chakraborty 2003].

The probability bounds Λ', Ω' of the output curves comes from the following lemma 4.2.1, but first we define the partial ordering among probabilistic curves.

Table 4.1: Probabilistic characteristic of residual service and arrival curves.

$\alpha(\Delta)$	$\beta(\Delta)$	$\beta'(\Delta)$	$\alpha'(\Delta)$
$\Omega = 0$	$\Lambda = 0$	$\Lambda' = 0$	$\Omega' = 0$
$\Omega = 0$	$0 < \Lambda \leq 1$	$\Lambda' = \Lambda$	$\Omega' = \Lambda$
$0 < \Omega \leq 1$	$\Lambda = 0$	$\Lambda' = \Omega$	$\Omega' = \Omega$
$0 < \Omega \leq 1$	$0 < \Lambda \leq 1$	$\Lambda' = \Omega + \Lambda - \Omega\Lambda$	$\Omega' = \Omega + \Lambda - \Omega\Lambda$

Definition ["Greater than or Equal to "($\succeq$)] The operator ($\succeq$) is defined over two probabilistic curves $\langle \omega, \Omega \rangle$ and $\langle \lambda, \Lambda \rangle$, with ω and λ the curves and Ω and Λ their respective bounding probabilities, as $\langle \omega, \Omega \rangle \succeq \langle \lambda, \Lambda \rangle \iff \omega \geq \lambda \wedge \Omega \leq \Lambda$.

Theorem 4.2.1 (Probability bound) *Given the arrival curve $\langle \alpha, \Omega \rangle$ and the service $\langle \beta, \Lambda \rangle$ of a component, the residual arrival curve $\langle \alpha', \Omega' \rangle$ and the residual service curve $\langle \beta', \Lambda' \rangle$ of a component have probability bound of $\Omega + \Lambda - \Omega\Lambda$. That is, $\Omega' = \Lambda' = \Omega + \Lambda - \Omega\Lambda$*

Proof From the definitions 4.2.1, 4.2.1, 4.2.2 and 4.2.2 we have $P[\mathbb{X}(t) \geq R(t)] \leq \Omega$ and $P[\mathbb{Y}(t) < C(t)] \leq \Lambda$. Let $P[A] = \Omega$ and $P[B] = \Lambda$ be the case when the definitions 4.2.1, 4.2.1, 4.2.2 and 4.2.2 are violated. Since, these two probabilities are not mutually exclusive (as both $R(t)$ and $C(t)$ can change simultaneously) and are independent, with the probability of $R(t)$ being larger and $C(t)$ being smaller equal to Ω and Λ respectively. Hence,

$$P[A \vee B] \quad = P[A] + P[B] - P[A]P[B] = \Omega + \Lambda - \Omega\Lambda,$$

For example, let $\langle \alpha, \Omega \rangle$ be an arrival curve such that $R(t) - R(s) \leq \alpha(t - s)$ and such that $P[\mathbb{X}(t) \geq R(t)] \leq \Omega$. Therefore, for some other curve $\langle \alpha^*, \Omega^* \rangle$ such that $\langle \alpha^*, \Omega^* \rangle \succeq \langle \alpha, \Omega \rangle$, the probability of $\langle \alpha^*, \Omega^* \rangle$ being larger is equal to Ω. Similarly, for some service curves we can reason that the probability $\langle \beta, \Lambda \rangle \succeq \langle \beta^*, \Lambda^* \rangle$ is given by Λ. Therefore, the probability bound of the variations in the residual curves, computed using the curves $\alpha*$ and $\beta*$, is equal to the probability of variation in either of the interfaces or both (given by Theorem 4.2.1).

Theorem 4.2.1 provides the probability bound for the curves at the output interface of a component. The theorem 4.2.1 can be summarized using the table 4.1, which gives the relationship between input and output probability bounds (assuming independence among inputs). There are four possible combinations of probability bounds for the two input curves $\langle \alpha, \Omega \rangle$ and $\langle \beta, \Lambda \rangle$. Where, as mentioned in previous section, probability bound equal to zero indicates the deterministic case[3]. We can now analyze component systems with a mix of deterministic and probabilistic components (i.e. the components with deterministic and probabilistic input interfaces) composing it (differentiated by the probabilistic bounds); this makes the analysis richer and suitable for better dimensioning.

[3] The occurrence of rare events can be handled using large deviation theory (see [Navet 2007]). By rare events we mean those events that have the probability of appearance close to zero.

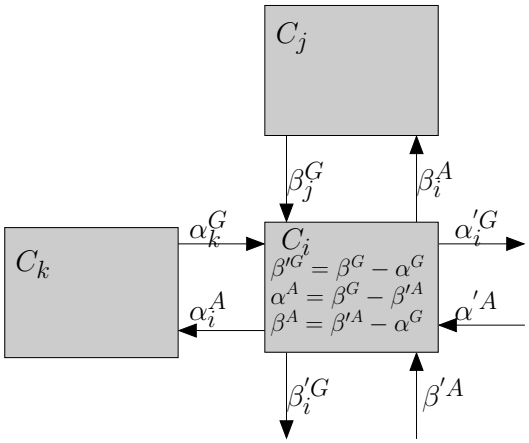


Figure 4.2: A component and its interface abstraction in the assume-guarantee form.

Lemma 4.2.2 (Max probability) *Given the arrival curve* $\langle \alpha, \Omega \rangle$ *and the service curve* $\langle \beta, \Lambda \rangle$ *of a component, the probability bounds* Ω' *and* Λ' *of the residual arrival curve* $\langle \alpha', \Omega' \rangle$ *and the residual service curves* $\langle \beta', \Lambda' \rangle$ *of a component is such that* Ω' *and* λ' *is larger than or equal to* $\max(\Lambda, \Omega)$.

Proof From Theorem 4.2.1 the residual probability bounds Ω' and Λ' is given by $\Omega + \Lambda - \Omega\Lambda$. The proof is given by contradiction by showing that following is not valid:

$$\Omega + \Lambda - \Omega\Lambda < \max(\Omega, \Lambda)$$

Assuming $\Omega = \max(\Omega, \Lambda)$, since both Ω and Λ are positive real number, we can subtract Ω from both sides of the equation (4.5) obtaining $\Lambda - \Omega\Lambda < 0$. Then, by adding $\Omega\Lambda$ to both sides of the former equation we get $\Lambda < \Omega\Lambda$ which is false as Ω cannot be greater than one.

From Lemma 4.2.2 we can conclude that the output probability bound of the curves either remains the same or increases, compared to the probability bound of the input curves.

4.3 Component-based probabilistic analysis

Henzinger et al. [Henzinger 2006] proposed assume-guarantee interfaces which are particular instances of real-time interfaces and consider a) the requirements of a component in terms of resource or expected arrivals in order to work properly, and b) the resource or arrivals a component provides. According to the assume-guarantee abstraction, in a real-time component-based system there is a component requesting for the computational resource and another component providing such resource [Thiele 2006]. For example in figure 4.2, a component i which schedules an application of tasks Γ_i, *assumes* a minimum amount of resource, β_i^A, in order to work properly and expects a maximum amount of work α_i^A, such that β_i^A is

enough to handle workload of the assumed work α_i^A by the component. A resource provisioning component j *guarantees* a minimum amount of resource, β_j^G. The load generating component k *guarantees* a maximum workload of α_k^G. The component i is compatible with the component k on its arrival interface if the workload generated by component k is less than or equal to the workload assumed by the component i, i.e. $\alpha^G \leq \alpha^A$. The reason being that if α^A can be scheduled by the component then so is α^G. Similarly, the component i is compatible with the component j if $\beta^G \geq \beta^A$. We can summarize these conditions into the predicate φ represents the assumptions on the arrival and service curves by the component and defines the composability among component as: $\varphi = \{(\alpha^G \leq \alpha^A) \wedge (\beta^G \geq \beta^A), (\beta'^A \leq \beta'^G)\}$.

4.3.1 Probabilistic interfaces

We now extend the component interface model to the probabilistic model.

Definition [Probabilistic interface] An interface with probability bound based probabilistic guarantees on inputs $\langle \alpha(\Delta), \Omega \rangle$ and $\langle \beta(\Delta), \Lambda \rangle$)(respectively the arrival and service curves), and on outputs $\langle \alpha'(\Delta), \Omega' \rangle$ and $\langle \beta'(\Delta), \Lambda' \rangle$ is the probabilistic interface.

The input/output interfaces are defined as:

$$\gamma = \langle \alpha, \Omega \rangle; \eta = \langle \beta, \Lambda \rangle; \gamma' = \langle \alpha', \Omega' \rangle; \eta' = \langle \beta', \Lambda' \rangle.$$

Definition [Probabilistic Component] Components that have probabilistic interfaces are probabilistic components. A probabilistic component C_i is defined as, $C_i = \{\gamma_i, \eta_i, \gamma_i', \eta_i'\}$.

In terms of assume-guarantee real-time interfaces, the probabilistic version, for the predicate φ becomes:

$$\begin{aligned} \varphi = \quad & \{\langle \alpha^G, \Omega \rangle \leq \langle \alpha^A, \Lambda_c \rangle \wedge \langle \beta^G, \Lambda \rangle \geq \langle \beta^A, \Lambda_c \rangle), \\ & \langle \beta'^A, \Lambda' \rangle \leq \langle \beta'^G, \Lambda' \rangle\} \end{aligned} \tag{4.5}$$

Where Λ_c probability threshold of the component, which will be used later as a safety threshold.

Definition [Degree of compatibility] Is the level of certainty with which interfaces of the two components are compatible (can be joined together) with each other, represented by the probabilistic value.

For example, in figure 4.2 for the components C_k and C_i if the assumed arrival curve is $\langle \alpha^A, 0 \rangle$ and the guaranteed arrival curve is $\langle \alpha^G, \Omega \rangle$ such that $\alpha^A \geq \alpha^G$. Therefore, the *degree of compatibility* on the interface between the two components is Ω. Which is intuitive since the guaranteed curve can be more only with the probability of Ω.

We can now analyze the requirements on the $\langle \alpha^G, \Omega \rangle \leq \langle \alpha^A, \Lambda_c \rangle$ and $\langle \beta^G, \Lambda \rangle \geq \langle \beta^A, \Lambda_c \rangle$ of the predicate, using following lemmas.

Lemma 4.3.1 *[Arrival Predicate] The degree of compatibility of the two components interfaces (see figure 4.2), i.e.* $\langle \alpha^G, \Omega \rangle \leq \langle \alpha^A, \Lambda_c = 0 \rangle$*, has the probabilistic bound of less than or equal to* Ω*.*

Proof In order to explain the requirements of the $\alpha^G \leq \alpha^A$, we can divide it into two parts $\alpha^G = \alpha^A$ and $\alpha^G < \alpha^A$. In the first case $\alpha^G = \alpha^A$ the arrival predicate will be true, but with a probability bound equal to Ω (as we may have a higher α^G with a probability of Ω) and for the second case the predicate will be true, but with a probability failure of less than Ω. See figure 4.3 for an explanation, $\Omega_1 < \Omega_2$ thus the upper α^G is more tighter and the probability of existence another tighter α^G than existing one will decrease as $\Omega < \Omega_1$, thus probability of failure for the predicate will be lesser than existing Ω.

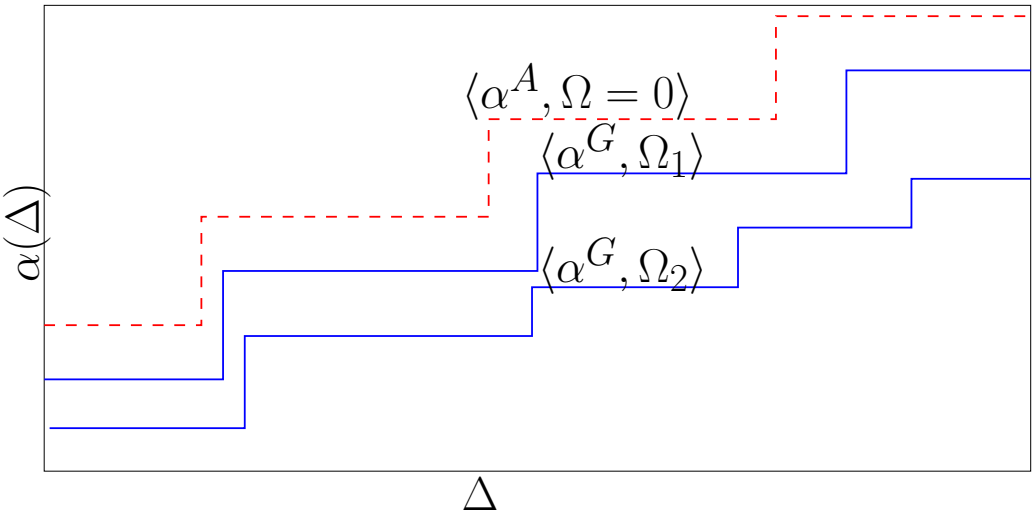


Figure 4.3: Comparison of the arrival curves with different probability bounds. The probability of α^G decreases going towards α^A and is zero for increasing beyond α^A, since $\Omega_1 < \Omega_2$.

Lemma 4.3.2 *[Service Predicate] The degree of compatibility of the two components interfaces (see figure 4.2), i.e.* $\langle \beta^G, \Lambda \rangle \geq \langle \beta^A, \Lambda_c = 0 \rangle$*, has the probabilistic bound of less than or equal to* Λ

Proof See Figure 4.4 and applying the reasoning as in lemma 4.3.1.

For the case when $\Lambda_c \neq 0$ the degree of composability is given by Theorem 4.2.1. Thus we see that the component interface composability acquires a richer meaning with the concept of *degree of compatibility*, than the idea of the concrete interface composability, which can help in better dimensioning of a system.

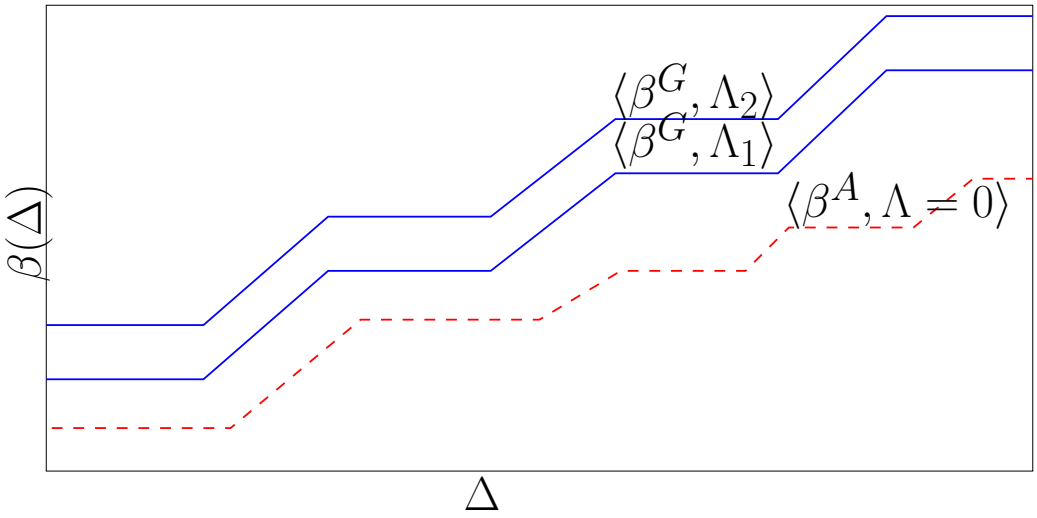


Figure 4.4: Comparison of the service curves with different values of probability bound. The probability of β^G decreases going towards β^A and is zero for decreasing beyond β^A, since $\Lambda_1 < \Lambda_2$.

4.3.2 Composability

In case of real-time systems the resource composability is equivalent to the classical schedulability criteria: *the resource provided to a component by another component has to be enough to satisfy the timing requirements of the component itself* [Thiele 2006, Wandeler 2006a, Wandeler 2005]. Two components are composable if all internal connections are compatible and if all open input predicates and all output predicates are still satisfiable.

The following theorem gives the notion of composability for component with probabilistic interfaces.

Theorem 4.3.3 *[Composability] The composability of components is guaranteed if* $\langle \alpha^G, \Omega \rangle \leq \langle \alpha^A, \Omega \rangle \wedge \langle \beta^G, \Lambda \rangle \geq \langle \beta^A, \Lambda \rangle)$ *holds*

.

Proof proof is a direct consequence of lemma 4.3.1 and 4.3.2 as predicate of Equation (4.5) which has to be satisfied in order to guarantee the composability.

The composability of the components is affected by the scheduling policy which defines the resource distribution among the components. In case of fixed priority scheduling the priority describes the composition order among the tasks. Figure 4.7 depicts a fixed priority (FP) scheduling, see [Lehoczky 1989], for n tasks each of them modeled as a component with assume-guarantee interface, where the β_is are the resources passed among the components.

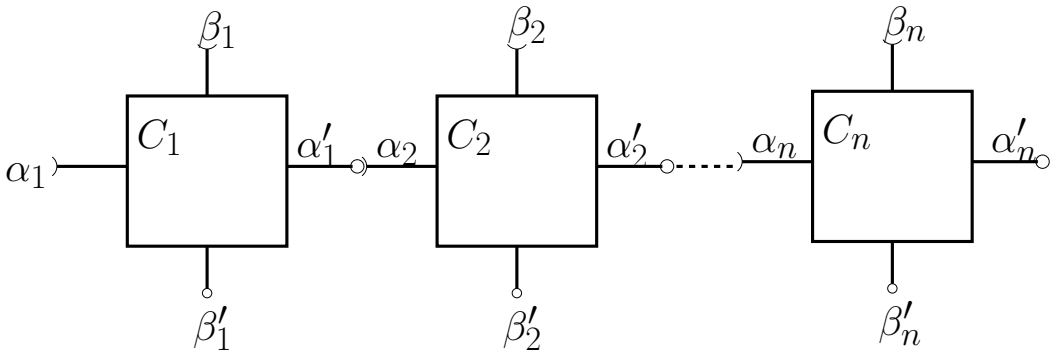


Figure 4.5: Example of an arrival chain of components.

The composition of components being served by a common arrival curve or service curve can be categorized into two classes such as *service chain* and *arrival chain*.

Definition [Service chain] A service chain, see figure 4.7, is a chain of components with one service curve going to first component of the chain and the remaining components being served by the residual service from previous component in the chain and all the components have different arrival curves.

Therefore, in a service chain $\{C_1, C_2, \ldots, C_k, \ldots C_n\}$ of n components with β_1 as an input service to C_1 and β'_1 as an input service to C_2 and likewise for rest of the components in the service chain. The probability bounds for the output interface of the component C_1 is $P_1 = \Omega_1 + \Lambda_1 - \Omega_1\Lambda_1$ (see Table 4.1) and the probability bound for the output interface of the component C_2 is $P_2 = P_1 + \Omega_2 - P_1\Omega_2$, since the input probability bound for β_2 is same as output probability bound of β'_1. The probability bound for the output interface of the component C_k in a service chain can be found using induction and is given by:

$$P_k = P_{k-1} + \Omega_k - P_{k-1}\Omega_k \tag{4.6}$$

Definition [Arrival chain] An arrival chain, see figure 4.5, is a chain of components with one arrival curve going to first component in the chain and the subsequent component receiving residual arrival curve and all the components in chain have their own service curves.

Therefore, in an arrival chain $\{C_1, C_2, \ldots, C_k, \ldots C_n\}$ of n components with α_1 as an input arrival to C_1 and α'_1 as an input arrival to C_2 and likewise for rest of the components in the arrival chain. The probability bound for the output interface of the component C_1 is $P_1 = \Omega_1 + \Lambda_1 - \Omega_1\Lambda_1$ (see Table 4.1) and the probability bound for the output interface of the component C_2 is $P_2 = P_1 + \Lambda_2 - P_1\Lambda_2$, since the input probability bound for α_2 is same as output probability bound of α'_1. Thus,

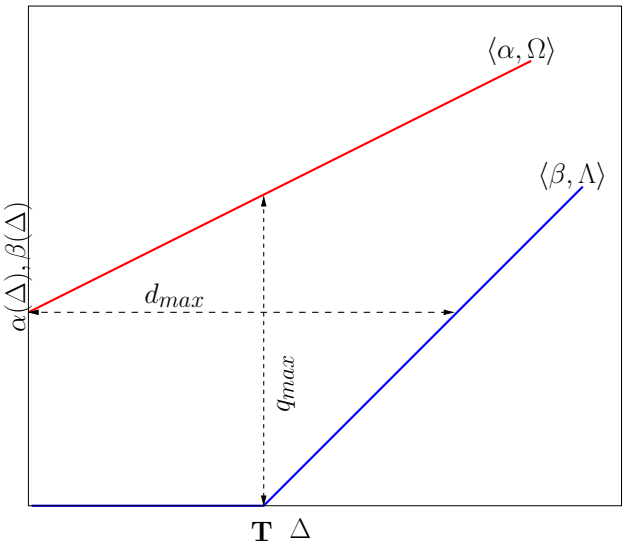


Figure 4.6: Computation of delay and backlog as maximum horizontal and vertical distance respectively.

the probability bound for the output interface of the component C_k in an arrival chain can be found using induction and is given by:

$$P_k = P_{k-1} + \Lambda_k - P_{k-1}\Lambda_k \tag{4.7}$$

The probability bound given for a component C_k in the service chain or the arrival chain is worst-case bound, as we stated in lemma 4.2.2.

4.3.3 Component system metrics

The real-time analysis applies *delays* (d) and *backlogs* (q) for schedulability purposes, see [Chakraborty 2003, Thiele 2000].

Delay.

Given an arrival curve and a service curve as input to a component, the maximum delay (or response time) experienced by an event given the resource represented by the service curves is the maximum number of backlogged events from the stream waiting to be processed, see figure 4.6, and can be given by the following inequalities [Chakraborty 2003]:

$$d_{max} \leq \sup_{\Delta \geq 0}\{\inf\{\gamma \geq 0 \ | \alpha^u(\Delta) \leq \beta^l(\Delta + \gamma)\}\} \tag{4.8}$$

Simply the delay is the maximum horizontal distance between the arrival curve and the service curves. Using the delay, it is possible to define the schedulability of task sets which depends on the scheduling policy applied as we have showed in the previous sections. Indeed, the delay is the amount of time that an application has to wait in order to have the necessary amount of resource available and then

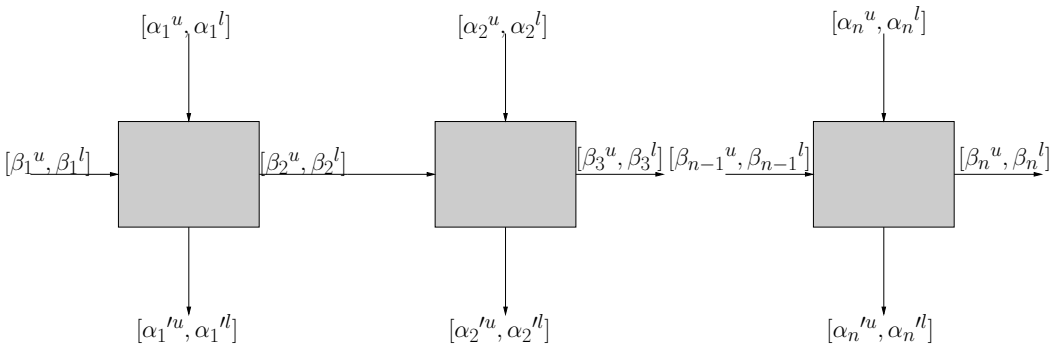


Figure 4.7: Example of fixed-priority scheduling: the service curve is passed according to the priority assignment form the highest priority component to the lowest priority one.

execute. If that delay is less than or equal to the component timing requirement (the deadline for task components), then the two component are composable, hence their applications are schedulable, otherwise not. The probability that an event has to wait for more than d_{max} delay before being processed is given by Λ' (equal to Ω'), as defined in Theorem 4.2.1.

Backlog.

On the other hand, the backlog q_{max} is the requirement of the component, given α and β as input, to avoid loss of data being unprocessed. It is the maximum vertical distance between the arrival curve and service curves (see figure 4.6), which gives the maximum number of events waiting to be served (thus need to be stored and hence gives the buffer requirement) and is given as [Chakraborty 2003]:

$$q_{max} \leq \sup_{\Delta \geq 0} \{\alpha^u(\Delta) - \beta^l(\Delta)\} \tag{4.9}$$

The probability that the available resource β dispatches the workload α before the backlog q_{max} overflows, is given by the probability Λ' (equal to Ω'), as defined in Theorem 4.2.1.

4.3.4 Schedulability

The schedulability of a component relies on the comparison among its input curves, the arrival and the service curve. In particular, sufficient condition can be derived by comparing the upper bound of the arrival curve and the lower bound of the service. Intuitively, whenever the arrival curve is lower than the service curve the component is schedulable, as we have enough service to handle the work.

With a probabilistic definition of curves, schedulability criteria can be extended in order to include the probability bounds. Thus, a flexible view of schedulability conditions can be inferred.

The composability criteria in case of Fixed Priority (FP) scheduling policies can be derived in a compositional manner [Chokshi 2008, Huang 2009, Wandeler 2005]. For the probabilistic component system we can summarize the FP schedulability condition as:

Theorem 4.3.4 *[FP Composability] A chain of FP components is composable with a resource provisioning component that guarantees $\langle \beta^G, \Lambda^G \rangle$ amount of resource if the demand from the highest priority component $\langle \beta_1^A, \Lambda_1^A \rangle$ is such that:*

$$\forall\ \Delta \quad \beta_1^A(\Delta) \leq \beta^G(\Delta), \quad \wedge \quad \Lambda_1^A \geq \Lambda^G \tag{4.10}$$

With β_1^A the resource assumed by highest priority component computed using Equation (4.11) and Λ_1 computed using Equation (4.6).

Proof Suppose we have n tasks (abstracted by a component having an arrival curve and service curve) in an application Γ. Without loss of generality, we assume the tasks to be an order set, according to their priorities, where τ_i is of higher than τ_k for $k > i$. Let $\{C_1, C_2, \ldots, C_k, \ldots C_n\}$ be the components abstracting the ordered set of tasks, i.e. a service chain. Suppose that $\langle \beta_1^l(\Delta), \Lambda_1 \rangle$ be the lower service curve provided to the highest priority component (i.e. service chain). The residual lower service curve $\langle \beta_1'(\Delta), \Lambda_1' \rangle$ after scheduling the highest priority component C_1 is computed using equation 4.3 and equation 4.6. In FP scheduling, the residual service is used to serve the next component in the service chain. Therefore, the assumed service $\langle \beta_n^A, \Lambda_n \rangle$ of the component C_n abstracting the task τ_n must be at least $\beta_n^A(\Delta) = \alpha_n^u(\Delta - D_n)$. Where D_n is the deadline constraint for the n-th task. Thus, the residual service curve β_{n-1}' after serving $n-1$ components in the service chain must be at least equal to $\beta_n^A(\Delta)$.

Therefore, the service bounds $\beta_{n-1}^A(\Delta), \beta_{n-2}^A(\Delta), \ldots, \beta_2^A(\Delta)$, can be computed sequentially. Knowing $\beta_k^A(\Delta)$, the bound $\beta_{k-1}^\sharp(\Delta)$ on β_{k-1}^l can be derived such that the residual service curve is guaranteed to be greater than or equal to $\beta_k^A(\Delta)$ if $\beta_{k-1}^l(\Delta)$ is greater than or equal $\beta_{k-1}^\sharp(\Delta)$:

$$\beta_{k-1}^\sharp(\Delta) = \beta_k^A(\Delta - \lambda) + \alpha_{k-1}^u(\Delta - \lambda) \tag{4.11}$$

where $\lambda = \sup\{\tau : \beta_k^A(\Delta - \tau) = \beta_k^A(\Delta)\}$. Furthermore, $\beta_{k-1}^l(\Delta)$ must be no less than $\alpha_{k-1}^u(\Delta - D_{k-1})$ to guarantee the constraint D_{k-1}. Therefore

$$\beta_{k-1}^A(\Delta) = \max\{\beta_{k-1}^\sharp(\Delta), \alpha_{k-1}^u(\Delta - D_{k-1})\}.$$

By applying the equation 4.11 for $k = n-1, n-2, \ldots, 2$, we can derive the lower service curve, i.e., $\beta_1^A(\Delta)$. From, equation 4.6 and theorem 4.2.1 for assumed interface we have $\Lambda_k^A \geq \Lambda_{k+1}^A$. Therefore, using lemma 4.3.2 we can say that if $\Lambda^G \leq \Lambda_1^A$ then guaranteed is stricter i.e. has lesser probability of decreasing below the assumed service.

4.4 Safety guarantees

The requirements on real-time guarantees is a mandatory characteristic for real-time components. In a mixed deterministic-probabilistic component system this is a challenging task since we have to provide a mechanism for giving quantitatively

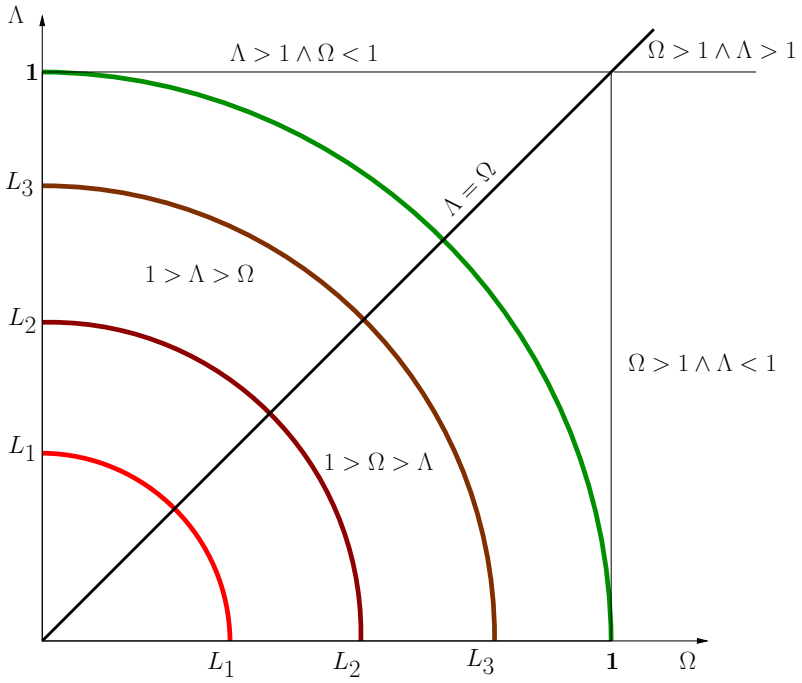


Figure 4.8: Level-L schedulability of a component based on SIL and identified using Ω and Λ: lower values of L mean higher safety. Where x-axis and y-axis represent arrival and service bounds respectively.

verifiable measure on components; this is a difficult task given the probabilistic nature of some components in system. Therefore, we require a measure which can quantify the degree to which requirements are met. Since, real-time system are mostly used in critical application like avionics, automotive etc., the safety seems to be a reasonable measure.

In order to provide measurable safety guarantees on the analysis we use SIL. For example, the SIL safety bound for a probabilistic component may be determined using methods described in [Gulland 2004].

Definition [Safety measure] The safety measure is the probability value associated to the components, such that the measure gives the confidence with which the component can be expected to perform its given function.

The *safety measure* can be a threshold associated to a component from the SIL standards, such that it guarantees that threshold (i.e probability bounds of all interfaces are less or equal to SIL threshold). Consider a component C_i with $\langle \beta_i, \Lambda_i \rangle$ as an input service curve and $\langle \alpha_i, \Omega_i \rangle$ as an input arrival curve. The probability bound for residual service and arrival curves for a component C_i is given by Theorem 4.2.1, which is $\Omega_i + \Lambda_i - \Omega_i \Lambda_i$. For such a component C_i the *safety measure* through the SIL can convey the idea of a *Level-L* composability and schedulability, with L defining the probability value for a threshold (i.e. safety-threshold requested

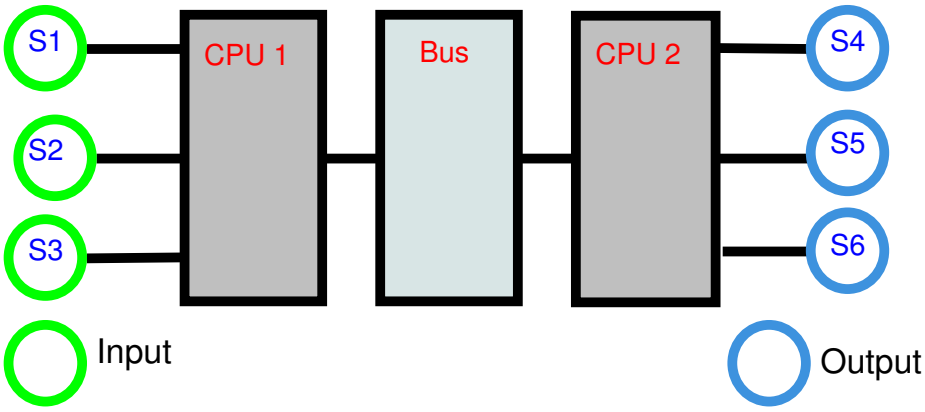


Figure 4.9: Case study: distributed system consisting of two CPUs joined by a bus.

or required) in a SIL standard (e.g. $IEC61508$).

Definition [Level-L Composability and Schedulability] The Level-L composability and schedulability is the *Safety measure* of the composability and the schedulability of a component, which gives the measure of the confidence with which the system can be expected to be composable and schedulable, where L is the probability value of a threshold in a SIL standard.

Therefore, the probability bounds (input and residual) of the component C_i, in order to be Level-L composabile and schedulable, translates into guaranteeing that the probability bounds are less than or equal to the level-L. Which implies the level-L component C_i. Thus, for the component C_i with $\langle \beta_i^G, \Lambda_i \rangle$ and $\langle \beta_i^G, \Omega_i \rangle$ as guaranteed curves and Λ_i and Ω_i less than equal to L in order to be classified as the level L component (composable and schedulable) the probability bound of the component's output interface should be bounded as:

$$\Omega_i + \Lambda_i - \Omega_i \Lambda_i \leq L. \tag{4.12}$$

The Figure 4.8 shows the regions of safety in a composition, where each axis represents the probability bounds of input service and arrivals and the semi-circular region gives the SIL level of the component after composition. After composing components the residual probabilities may move to a higher SIL region for a component (depending on the values of input probability bounds), which means lower guarantees for the component or a lower schedulability. The reason being that the value of probability bound increases after composition, that is what Theorem 4.2.1 and Lemma 4.2.2 tell us.

Example For a component C having input probability bound(for both input interfaces) equal to Λ, the probability bound for the output interface of the component should be $2\Lambda - \Lambda^2 - L \leq 0$, in order for C to be called as Level-L SIL component. Conversely, we can say that the input probabilities should be bound as $\Lambda \leq 1 \pm \sqrt{1 - L}$ for a component to be level-L component.

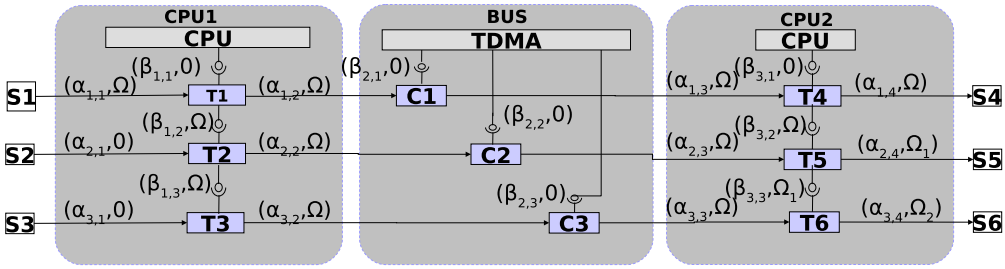


Figure 4.10: Case study: component architecture representation with interface and probabilistic curves applied. Where $\Omega_1 = 2\Omega - \Omega^2$ and $\Omega_2 = 3\Omega - 3\Omega^2 + \Omega^3$, computed using Theorem (4.2.2).

Table 4.2: Input streams (tasks) specification of the distributed system.

Stream	*Parameters*	D	Task Chain
$\alpha_{1,1}$	$\Omega = \{10^{-4} - 10^{-6}\}$	48	$T1 \rightarrow c_1 \rightarrow T4$
$\alpha_{2,1}$	$p = 10, j = 0, d = 10$	20	$T2 \rightarrow c_2 \rightarrow T5$
$\alpha_{3,1}$	$p = 10, j = 0, d = 10$	23	$T3 \rightarrow c_3 \rightarrow T6$

Thus for a SIL-L component the input and output probability bounds should be less than or equal to L, i.e. the probability bounds should stay within the semi-circular region of radius L. The composability, schedulability acquire a richer definition within probabilistic scenarios, as the probability bounds offer different degrees of composability, hence schedulability, among the components.

4.5 Case study

To analyze our case study depicted in Figure 4.9, we use Modular Performance Analysis (MPA) toolbox in MATLAB as user front-end, see [Wandeler 2006b].

The case study considers a distributed real-time system with 2 CPU's that communicate via shared bus, as in Figure 4.9. There are three input streams $S1$, $S2$ and $S3$ processed by chains of tasks. For example, the events of stream $S1$ are first processed by task $T1$ and the resulting stream is then processed by $T4$. The communication of the intermediate stream through the bus resource is modeled by a communication tasks $C1$, $C2$ and $C3$. The tasks $T1$, $T2$ and $T3$ are mapped to $CPU1$ and are scheduled according to Fixed Priority Non-Preemptive (FPNP) scheduling, with $T1$ having highest priority and $T3$ having lowest priority. Similarly, $T4$, $T5$ and $T6$ are mapped into $CPU2$ and scheduled according to FPNP scheduling, with $T4$ having highest priority and $T6$ having lowest priority. The computational requirement of each task is exactly 1 time unit. The bus uses Time Division Multiple Access (TDMA), where each communication task $C1$, $C2$ and $C3$ is periodically allocated the communication resource for 5 time units. For detailed specification of system architecture see Figure 4.10. The specification of the input

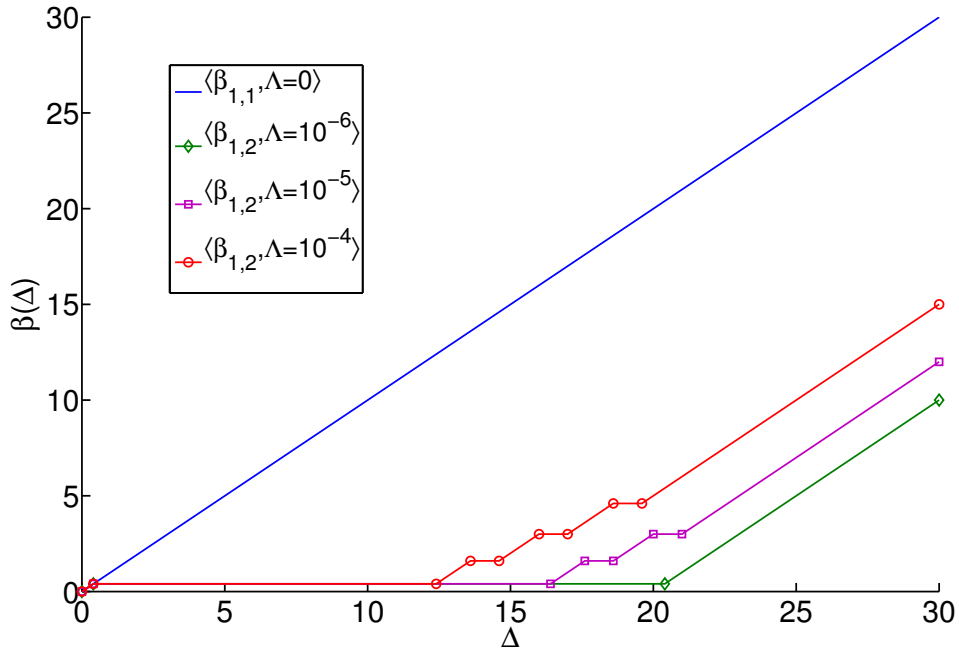


Figure 4.11: Input service curve and residual service curves for different values of Λ.

event streams is given in Table 4.2.

To generate AAC curve $\alpha_{1,1}$, for different probability bounds, we use the Weibull distribution and resulting curves are shown figure 4.12. The generated curves are then transformed to interface with the MPA toolbox, using a wrapper which takes into account the probability bounds.

The service curves $\beta_{1,1}, \beta_{2,1}, \beta_{2,2}, \beta_{2,2}, \beta_{2,3}, \beta_{3,1}$ are deterministic and thus have probability bound value of $\Lambda = 0$ and similarly $\alpha_{2,1}$ and $\alpha_{3,1}$ have probability bound value of $\Omega = 0$. In this case study we assume some of the arrivals with deterministic bounds, while others with probabilistic bounds (with bounds Ω different than 0) in order to motivate the flexibility of analyzing mixed (probabilistic and deterministic) components using the framework. Nevertheless, our approach can effectively work with complete deterministic or probabilistic systems. For the components receiving mixed inputs the output curves are computed using MPA and the probability bounds are computed using the Theorem 4.2.1.

For example, component $T1$ receives deterministic $\beta_{1,1}$ and AAC $\alpha_{1,1}$ with probability bound Ω varying between $10^{-4} - 10^{-6}$. The residual curves $\beta_{1,2}$ is computed using MPA, as can be seen in Figure 4.11. It has a probability bound value computed using the Theorem 4.2.1.

The figure 4.13 shows the input and output curves (arrival and service). The impact of ACC (α_{11}) becomes obvious after the initial events streams α_{21} and α_{31}, which are periodic and deterministic, show a much larger degree of non-determinism (upper and lower curves have a large distance) in the corresponding residual output

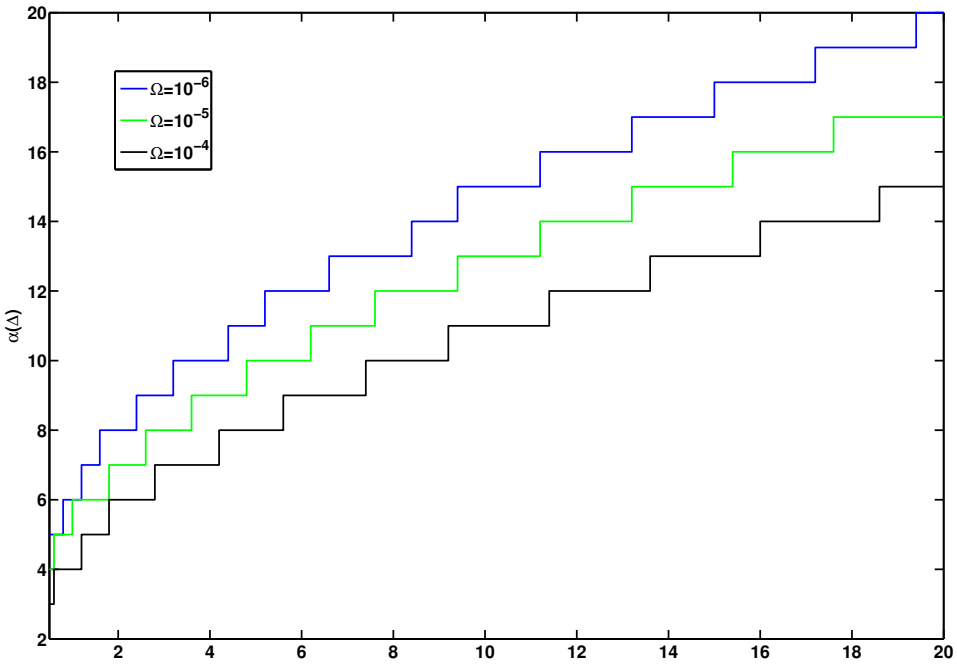
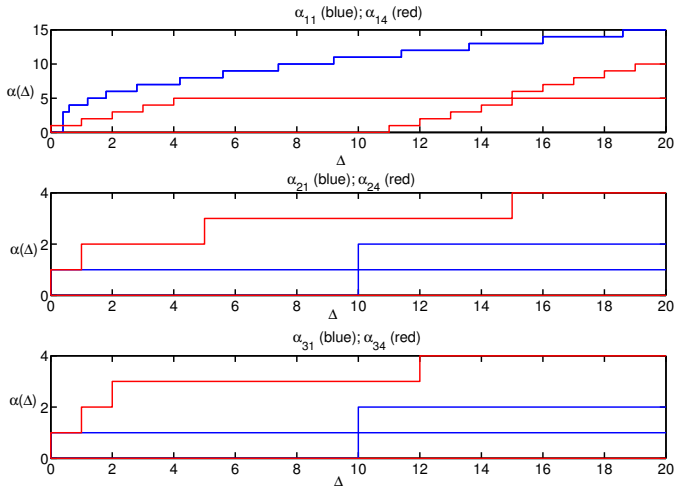


Figure 4.12: AAC curve $\alpha_{1,1}$, for different probability bounds.

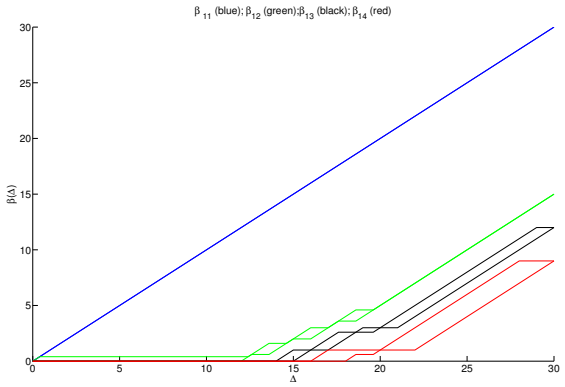
streams .

A closer look at the residual curves reveals the minimal interval between two subsequent events is the time interval (minimum) when the upper curve acquires value 2. Similarly, the largest interval between two subsequent events is the time interval (minimum) when the lower curve has value 1. In our case we find the intervals to be [1, -] for the residual events streams α_{24} and α_{34}, which are of course much larger variances than [10, 10] for the corresponding input event streams α_{21} and α_{31} . The service curves β_{11} and β_{31} represent the full service available from CPU1 and CPU2. β_{12}, β_{13} and β_{14} show the service available after fixed priority scheduler has allocated resources for tasks T1, T2 and T3, and it can be seen that not much is left in terms of available service. The changes to probability bound affects the α_{11} which in turn produces an effect of reduced available service for successive components. In Figure 4.11 it can be clearly seen that as the criticality/safety of $\alpha_{1,1}$ increases, the criticality of $\beta_{1,2}$ also increases resulting in reduced service offerings to the next component.

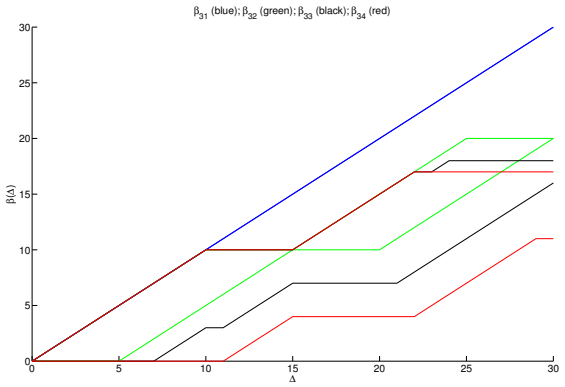
Deadline miss Given an arrival curve and a service curve as input to a component, we can compute the maximum delay for each component. Then, the delays of each component is composed to find the end-to-end delay, to find the schedulability of task chain with respect to deadlines given in Table 4.2. By comparing the delays and the deadlines (the maximum affordable delays) it is possible to conclude about the schedulability of the component or chain of components. For example, for task chain 1 the end-to-end delay changes from 36.4 to 49 as the criticality/safety level



(a) Arrival (blue) and residual (red) curves with on CPU1.



(b) Available service (blue) and residual service (red,green black) on CPU1.



(c) Available service (blue) and residual service (red,green black) on CPU2.

Figure 4.13: Results of analysis for the given case study.

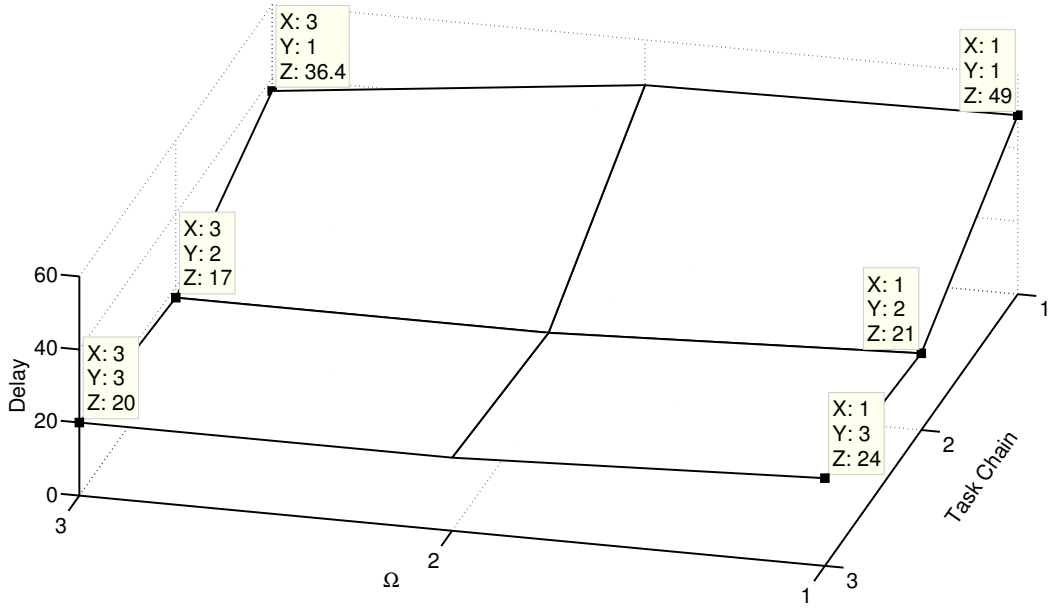


Figure 4.14: End-to-end delay of three task chains for different values of Ω. Where $\Omega = 1, 2$, and 3 corresponds to $10^{-6}, 10^{-5}$, and 10^{-4} respectively.

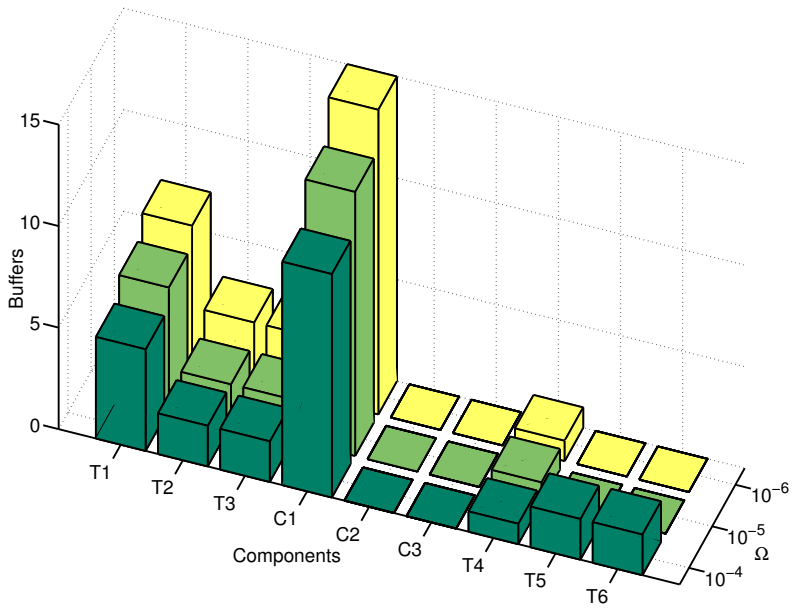


Figure 4.15: Buffers requirements w.r.t to probability bound on the input arrival curves.

changes from $10^{-4} - 10^{-6}$ for $\alpha_{1,1}$. As a result, it can be clearly seen that for higher criticality/safety-threshold the deadline requirements specified for task chains $1, 2, 3$ are not met, as can be seen in Figure 4.14. Following similar reasoning (that of end-to-end delay) it is possible to show how probability bound affect backlog, as is shown is figure 4.15. Therefore, it is interesting to such problems as we are now able to evaluate the response of a component systems with mixed interfaces (i.e. both deterministic and probabilistic event streams), with the different criticality/safety requirements.

4.6 Summary

In this chapter we have developed an analysis framework for component-based real-time systems. We have first defined a probabilistic version of the component interfaces based on bounds and probabilistic thresholds, through which it becomes possible to model both deterministic and probabilistic components. The resulting feasibility analysis is able to cope with mixed (probabilistic and deterministic) component systems where probabilistic and deterministic components interact. The framework is flexible enough to deal with a) incomplete specifications, as it can arise early in the design cycle, b) with different feasibility requirements: from hard real-time, requiring deterministic bounds, to soft real-time where probabilistic guarantees are enough, and c) allows better dimensioning of the system as we do not put any pessimistic conditions or assumptions of the resource demand or work arrivals.

In future works, we intend to apply the proposal to large distributed applications, such as automotive and avionic systems, and evaluate the outcomes in terms of complexity, tightness and expressiveness with regards to the other existing formalisms. Moreover, exploring other scheduling policies, than FP scheduling, can be taken care of with similar reasoning. Also would like to extend this framework so that it can handle and evaluate the occurrence of rare events, for instance through large deviations or importance sampling.

Chapter 5

Summary

Contents

This thesis presents a schedulability analyses for automotive systems and embedded networks, with the aim to facilitate cost-effective and reliable design and analysis of automotive embedded systems. The framework is applied in the automotive domain, so a to dimension the system better and to reduce the risk of deadline failure due to hardware limitations and interference due to probabilistic traffic. The analyses is shown to facilitate safety-criticality and flexible integration of probabilistic traffic into system modeling.

We began in Chapter 1 with the problem definition and the understanding of analyses requirements in the automotive embedded systems. We looked into the state of the art and presented limitations in terms of lack of modeling details such as, integrating hardware limitations, implementation overheads, safety and integration of aperiodic arrivals. This allowed us to understand the key points that need to be integrated into the analyses of the automotive embedded systems, which could result in the better system dimensioning of the system.

In Chapter 2, we developed a new approach for integrating the aperiodic traffic in response time analysis. The main interest of the proposal is that the overestimation of the aperiodic traffic is kept to the minimum that still enables the system to meet some chosen dependability requirements. The analysis developed can be pessimistic especially for lower priority frames when there is a large volume of aperiodic traffic, as we have assumed worst-case arrival process when estimating the release times from data trace. The estimated arrival process is burst in nature and will be seen more by the lower priority frames. It is possible to be less pessimistic by modeling each aperiodic stream individually and integrate only the higher priority aperiodic WAFs into the schedulability analysis. However, we believe that this more fine-grained approach will not be always practical since it requires significant modeling efforts and large quantity of data traces. We have provided few schemes which would minimize the pessimism due to priority issues and still respecting the safety threshold while being as accurate as possible (*i.e.*, discard as much as possible of the lower priority aperiodic traffic).

In Chapter 3, we gave an analytical model for schedulability analysis for CAN controllers when finite copy-time of messages is considered and when the transmis-

sion buffers can not be aborted. The models developed in this chapter provides very important understanding of the consequences due architectural limitations in CAN. We also derived a more realistic response time analysis in a typical case where controllers have three or more transmission buffers and the ability to cancel transmission requests is absent. As seen in case study of section 3.4 the implementation quality and the architecture of the CAN device driver can have consequences on the WCRT of messages and we provide the some guidelines to avoid the same. This analysis is of particular interest to automotive sector where multiple Tier 1 suppliers provide ready to use ECUs in an automobile. And the lack of knowledge at the time of integration, about the limitations of CAN controller used or device driver provided by tier 1 suppliers, can have serious consequences.

In chapter 4, we developed an analysis framework for component-based real-time systems. We first defined a probabilistic version of the component interfaces based on bounds and probabilistic thresholds, through which it becomes possible to model both deterministic and probabilistic components. The resulting feasibility analysis is able to cope with a systems with both probabilistic and deterministic arrivals. The framework is flexible enough to deal with a) incomplete specifications, as it can arise early in the design cycle, and b) with different feasibility requirements: from hard real-time, requiring deterministic bounds, to soft real-time where probabilistic guarantees are enough.

5.1 Future work

In chapter 2, the results hold under the assumption that the aperiodic inter-arrivals are independent and identically distributed. In practice, this assumption can be easily tested using statistical tests such as the BDS test (Brock, Dechert, Scheinkman) statistics but it is clear that it may not hold for all kinds of systems and workloads. Future work should be devoted to studies aimed at determining a schedulability analysis, in presence of non-i.i.d aperiodic load. It would be also interesting to study, for instance by simulation, how departure from the i.i.d. property impacts the accuracy of the results. Furthermore, it is interesting to include the corner cases in tailed distributions, perhaps through theory of large deviation.

In chapter 3, As seen in the case-study of section 3.6 the choice of priorities has an effect such that the additional delay gets reduced,therefore as a future work it would be very interesting to study the priority mapping schemes which could reduce the amount of additional delay in case a message suffers from priority inversion. Also, we will study the choice of offsets on ECUs so that messages are not released at the very same moment, to reduce the chances of priority inversion in a CAN controller. Moreover, the analysis should be extended for an arbitrary deadline case, with the effects of copy-time considered.

In Chapter 4, we intend to apply the proposal to large distributed applications, such as automotive and avionic systems, and evaluate the outcomes in terms of complexity, tightness and expressiveness with regards to the other existing formalisms.

Moreover, exploring other scheduling policies should be taken care of, with the same reasoning. We would like to extend this framework so that it can handle and evaluate the occurrence of rare events, for instance through large deviations or importance sampling. It would be interesting to apply this framework to a real case-study and then demonstrate its expressiveness.

5.1.1 Near Future

In near future I would like to achieve following milestones for this work:

- Develop a probabilistic model of aperiodic traffic arrivals, when we have tailed distributions and non-i.i.d cases.
- Develop a priority assignment algorithm for the system with probabilistic and deterministic arrivals, e.g. based on expectations.
- Develop a robust priority assignment algorithm that takes into account priority-inversion and resulting additional delay.
- Develop a Matlab based modeling and analyses toolbox for mixed (probabilistic and deterministic) component system.

List of Figures

List of Tables

Bibliography

[AUTOSAR 2009] AUTOSAR. *Specification of CAN Driver*. Autosar Release 4.0 Rev1. Available at http://www.autosar.org, 2009. (Cited on pages 40 and 42.)

[Bernat 2002] G. Bernat, A. Colin and S.M. Petters. *WCET analysis of probabilistic hard real-time systems*. In Real-Time Systems Symposium, 2002. RTSS 2002. 23rd IEEE, pages 279 – 288, 2002. (Cited on page 8.)

[Braun 2007] C. Braun, L. Havet and N. Navet. *NETCARBENCH: A Benchmark for Techniques and Tools Used in the Design of Automotive Communication Systems*. In 7th IFAC International Conference on Fieldbuses and Networks in Industrial and Embedded Systems, pages 321–328, 2007. (Cited on pages 33, 52 and 64.)

[Broock 1996] W.A. Broock, J.A. Scheinkman, W.D. Dechert and B. LeBaron. *A test for independence based on the correlation dimension*. Econometric Reviews, vol. 15, no. 3, pages 197–235, 1996. (Cited on pages 17 and 20.)

[Brumback 1987] B. Brumback and M. Srinath. *A Chi-Square test for fault-detection in Kalman filters*. Automatic Control, IEEE Transactions on, vol. 32, no. 6, pages 552–554, Jun 1987. (Cited on page 21.)

[Buck 2002] Joseph Buck, Soonhoi Ha, Edward A. Lee and David G. Messerschmitt. *Readings in hardware/software co-design*. chapitre Ptolemy: a framework for simulating and prototyping heterogeneous systems, pages 527–543. Kluwer Academic Publishers, 2002. (Cited on page 5.)

[Burns 2003] A. Burns, G. Bernat and I. Broster. *A probabilistic framework for schedulability analysis*. In Proceedings of the Third International Conference on Embedded Software (EMSOFT 2003), pages 1–15, 2003. (Cited on pages 13 and 69.)

[Chakraborty 2003] S. Chakraborty, S. Künzli and L. Thiele. *A General Framework for Analysing System Properties in Platform-Based Embedded System Designs*. In DATE, pages 190–195, 2003. (Cited on pages 68, 69, 72, 74, 81 and 82.)

[Chokshi 2008] Devesh B. Chokshi and Purandar Bhaduri. *Modeling Fixed Priority Non-Preemptive Scheduling with Real-Time Calculus*. In 14th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications (RTCSA 2008), pages 387–392, August 2008. (Cited on pages 72 and 82.)

[Cucu 2006] L. Cucu and E. Tovar. *A Framework for Response Time Analysis of Fixed-Priority Tasks with Stochastic Inter-arrival Times.* ACM SIGBED Review, vol. 3, no. 1, 2006. (Cited on page 69.)

[Davis 2007] R.I. Davis, A. Burn, R.J. Bril and J.J. Lukkien. *Controller Area Network (CAN) schedulability analysis: Refuted, revisited and revised.* Real-Time Systems, vol. 35, pages 239–272, 2007. (Cited on pages 3, 32, 40, 42, 50, 51, 53 and 59.)

[Davis 2011a] R.I. Davis, S. Kollmann, V. Pollex and F. Slomka. *Controller Area Network (CAN) Schedulability Analysis with FIFO queues.* In 23rd Euromicro Conference on Real-Time Systems (ECRTS), pages 45–56, 5-8th July 2011. (Cited on pages 41 and 47.)

[Davis 2011b] Robert Davis and Alan Burns. *Improved priority assignment for global fixed priority pre-emptive scheduling in multiprocessor real-time systems.* Real-Time Systems, vol. 47, pages 1–40, 2011. (Cited on page 62.)

[de Alfaro 2001] L. de Alfaro and T. Henzinger. *Interface Theories for Component-base Design.* In In EMSOFT'01: Embedded Software, Lecture notes in Computer Science 2211, pages 148–165. Springer Verilag, 2001. (Cited on page 69.)

[de Alfaro 2005] L. de Alfaro and T. Henzinger. *Interface-Based Design.* In To appear in the proceedings of the Marktoberdorf Summer School, 2005. (Cited on page 69.)

[Díaz 2002] José Luis Díaz, Daniel F. García, Kanghee Kim, Chang-Gun Lee, Lucia Lo Bello, José María López, Sang Lyul Min and Orazio Mirabella. *Stochastic Analysis of Periodic Real-Time Systems.* In RTSS '02: Proceedings of the 23rd IEEE Real-Time Systems Symposium, page 289, Washington, DC, USA, 2002. IEEE Computer Society. (Cited on pages 8 and 69.)

[Easwaran 2006] A. Easwaran, I. Shin, O. Sokolsky and I. Lee. *Incremental Schedulability Analysis of Hierarchical Real-Time Components.* In Proceedings of the 6th ACM & IEEE International Conference on Embedded Software (EMSOFT 2006), pages 272–281, October 2006. (Cited on page 69.)

[Gardner 1999] Mark Gardner and Jane Liu. *Analyzing Stochastic Fixed-Priority Real-Time Systems.* In W. Cleaveland, editeur, Tools and Algorithms for the Construction and Analysis of Systems, volume 1579 of *Lecture Notes in Computer Science*, pages 44–58. Springer Berlin / Heidelberg, 1999. (Cited on page 8.)

[Gonzalez Harbour 2001] M. Gonzalez Harbour, J.J. Gutierrez Garcia, J.C. Palencia Gutierrez and J.M. Drake Moyano. *MAST: Modeling and analysis suite for real time applications.* In Real-Time Systems, 13th Euromicro Conference on, 2001., pages 125 –134, 2001. (Cited on page 5.)

[Grenier 2008] M. Grenier and N. Navet. *Fine-Tuning MAC-Level Protocols for Optimized Real-Time QoS.* IEEE Transactions on Industrial Informatics, vol. 4, no. 1, pages 6 –15, February 2008. (Cited on page 40.)

[Gulland 2004] W G Gulland. *Methods of Determining Safety Integrity Level (SIL) Requirements - Pros and Cons.* In Practical Elements of Safety, Proceedings of the Twelfth Safety-critical Systems Symposium, Birmingham, UK, 17-19 February 2004. (Cited on page 84.)

[Hansson 2002] H.A. Hansson, T. Nolte, C. Norstrom and S. Punnekkat. *Integrating Reliability and Timing Analysis of CAN-Based Systems.* IEEE Transactions on Industrial Electronics, vol. 49, no. 6, pages 1240–1250, December 2002. (Cited on pages 41 and 68.)

[Henia 2005] R. Henia, A. Hamann, M. Jersak, R. Racu, K. Richter and R. Ernst. *System level performance analysis - the SymTA/S approach.* Computers and Digital Techniques, IEE Proceedings -, vol. 152, no. 2, pages 148 – 166, March 2005. (Cited on page 7.)

[Henriksson 2003] D. Henriksson, A. Cervin and K.E. Årzén. *TrueTime: Real-time control system simulation with MATLAB/Simulink.* In Proceedings of the Nordic MATLAB Conference, Copenhagen, Denmark, 2003. (Cited on page 5.)

[Henzinger 2006] T.A. Henzinger and S. Matic. *An Interface Algebra for Real-Time Components.* In Proceedings of the 12th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS'06), pages 253–266, April 2006. (Cited on pages 7 and 76.)

[Huang 2009] Kai Huang, Luca Santinelli, Jian-Jia Chen, Lothar Thiele and Giorgio C. Buttazzio. *Periodic Power Management Schemes for Real-Time Event Streams.* In the 48th IEEE Conf. on Decision and Control (CDC), pages 6224–6231, Shanghai, China, 2009. (Cited on page 82.)

[Jiang 2006] Yuming Jiang. *A basic stochastic network calculus.* SIGCOMM Comput. Commun. Rev., vol. 36, no. 4, pages 123–134, 2006. (Cited on page 70.)

[Jiang 2008] Yuming Jiang. Stochastic network calculus. Springer, London, 2008. (Cited on page 7.)

[Khan 2009] Dawood Khan, Nicolas Navet, Bernard Bavoux and Jörn Migge. *Aperiodic traffic in response time analyses with adjustable safety level.* In 14th IEEE International Conference on Emerging Techonologies and Factory Automation - ETFA, 2009. (Cited on page 26.)

[Khan 2010] Dawood A. Khan, Reinder J. Bril and Nicolas Navet. *Integrating Hardware Limitations in CAN Schedulability Analysis.* In Wip paper at the 8th

IEEE International Workshop on Factory Communication Systems (WFCS 2010), pages 207–210, May 2010. (Cited on pages 41 and 63.)

[Khan 2011] Dawood Khan, Davis Robert I. and Nicolas Navet. *Schedulability Analysis of CAN with Non-abortable Transmission Requests.* In 16th IEEE International Conference on Emerging Techonologies and Factory Automation - ETFA, 2011. (Cited on pages 48 and 63.)

[Le Boudec 2001] J. Y. Le Boudec and P. Thiran. Network calculus: A theory of deterministic queuing systems for the internet. Springer-Verlag New York, Inc., 2001. (Cited on pages 7, 69, 72 and 74.)

[Lehoczky 1989] John P. Lehoczky, Lui Sha and Y. Ding. *The Rate Monotonic Scheduling Algorithm: Exact Characterization and Average Case Behavior.* In IEEE Real-Time Systems Symposium, pages 166–171, 1989. (Cited on page 79.)

[Lorente 2006] Jose' L. Lorente, Giuseppe Lipari and Enrico Bini. *A Hierarchical Scheduling Model for Component-Based Real-Time Systems.* In Proc. of IPDPS'06, 2006. (Cited on pages 68, 69 and 71.)

[López 2008] J.M. López, J.L. Díaz, J Entrialgo and D. Garcia. *Stochastic Analysis of Real-Time Systems under Preemptive Priority-Driven Scheduling.* Journal of Real-time Systems, vol. 40, no. 2, 2008. (Cited on page 69.)

[Marti and 2010] P. Marti and, A. Camacho, M. Velasco and M. El Mongi Ben Gaid. *Runtime Allocation of Optional Control Jobs to a Set of CAN-Based Networked Control Systems.* IEEE Transactions on Industrial Informatics, vol. 6, no. 4, pages 503–520, November 2010. (Cited on page 40.)

[McShane 2008] Blake McShane, Moshe Adrian, Eric T Bradlow and Peter S Fader. *Count Models Based on Weibull Interarrival Times.* Journal of Business and Economic Statistics, vol. 26, no. 3, pages 369–378, July 2008. (Cited on page 26.)

[Meschi 1996] A. Meschi, M. Di Natale and M. Spuri. *Priority Inversion at the Network Adapter when Scheduling Messages with Earliest Deadline Techniques.* In 8th Euromicro Workshop on Real-Time Systems, pages 243–248, June 1996. (Cited on pages 41, 46 and 47.)

[Millard 1967] J. Millard and L. Kurz. *The Kolmogorov-Smirnov tests in signal detection (Corresp.).* IEEE Transactions on Information Theory, vol. 13, no. 2, pages 341–342, Apr 1967. (Cited on page 21.)

[Mok 2001] Aloysius K. Mok, Xiang (Alex) Feng and Deji Chen. *Resource Partition for Real-Time Systems.* In Real-Time Systems, 2001. (Cited on page 68.)

[Natale 2006] Marco Di Natale. *Evaluating message transmission times in Controller Area Networks without buffer preemption.* In 8th Brazilian Workshop on Real-Time Systems, 2006. (Cited on pages 40, 41, 60, 63 and 98.)

[Natale 2008] Marco Di Natale. *Understanding and using the Controller Area Network.* Handout of a lecture at U.C. Berkeley available at `http://inst.eecs.berkeley.edu/~ee249/fa08/`, October 2008. (Cited on page 41.)

[Navet 1998] Nicolas Navet and Ye-Qiong Song. *Design of Reliable Real-Time Applications Distributed over CAN (Controller Area Network).* In INCOM98, IFAC 9th Symposium on Information Control in Manufacturing, page 6 p, 1998. (Cited on page 69.)

[Navet 2000] N. Navet, Y.Q. Song and F. Simonot. *Worst-case deadline failure probability in real-time applications distributed over Controller Area Network(CAN).* Journal of Systems Architecture, vol. 46, pages 607–617, 2000. (Cited on pages 7 and 69.)

[Navet 2005] Nicolas Navet, Ye-Qiong Song, Françoise Simonot Lion and Cédric Wilwert. *Trends in Automotive Communication Systems.* Proceedings of the IEEE, vol. 93, no. 6, pages 1204–1223, Jun 2005. (Cited on page 40.)

[Navet 2007] N. Navet, L. Cucu and R. Schott. *Probabilistic Estimation of Response Times Through Large Deviations.* In WiP of 28th IEEE Real-Time Systems Symposium (RTSS'2007 WiP). IEEE, 2007. (Cited on pages 13 and 75.)

[Nilsson 2009] Dennis Nilsson, Ulf Larson, Francesco Picasso and Erland Jonsson. *A First Simulation of Attacks in the Automotive Network Communications Protocol FlexRay.* In Emilio Corchado, Rodolfo Zunino, Paolo Gastaldo and Alvaro Herrero, editeurs, Proceedings of the International Workshop on Computational Intelligence in Security for Information Systems CISIS'08, volume 53 of *Advances in Intelligent and Soft Computing*, pages 84–91. Springer Berlin / Heidelberg, 2009. (Cited on page 6.)

[Nolte 2001] T. Nolte, H. Hansson, C. Norstrom and S. Punnekkat. *Using bit-stuffing distributions in CAN analysis.* In IEEE/IEE Real-Time Embedded Systems Workshop (Satellite of the IEEE Real-Time Systems Symposium) London, 2001. (Cited on page 8.)

[Pop 2002] Traian Pop, Petru Eles and Zebo Peng. *Holistic scheduling and analysis of mixed time/event-triggered distributed embedded systems.* In Proceedings of the tenth international symposium on Hardware/software codesign, CODES '02, pages 187–192, New York, NY, USA, 2002. ACM. (Cited on page 6.)

[rts] *RTaW-Sim.* (Cited on page 5.)

[Santinelli 2011] Luca Santinelli, Patrick Meumeu Yomsy, Dorin Maxim and Liliana Cucu-Grosjean. *A Component-Based Framework for Modeling and Analysing Probabilistic Real-Time Systems.* In 16th IEEE International Conference on Emerging Technologies and Factory Automation, 2011. (Cited on page 70.)

[Shin 2003] Insik Shin and Insup Lee. *Periodic Resource Model for Compositional Real-Time Guarantees.* In RTSS '03: Proceedings of the 24th IEEE International Real-Time Systems Symposium, page 2, Washington, DC, USA, 2003. IEEE Computer Society. (Cited on pages 68 and 71.)

[Shin 2004a] Insik Shin and Insup Lee. *A Compositional Framework for Real-Time Guarantees.* In ASWSD, pages 43–56, 2004. (Cited on page 69.)

[Shin 2004b] Insik Shin and Insup Lee. *Compositional Real-Time Scheduling Framework.* In 25th IEEE International Real-Time System Symposium, pages 57–67, 2004. (Cited on pages 68 and 69.)

[Shin 2008a] Insik Shin, Moris Behnam, Thomas Nolte and Mikael Nolin. *Synthesis of Optimal Interfaces for Hierarchical Scheduling with Resources.* Real-Time Systems Symposium, IEEE International, vol. 0, pages 209–220, 2008. (Cited on page 69.)

[Shin 2008b] Insik Shin and Insup Lee. *Compositional real-time scheduling framework with periodic model.* ACM Trans. Embed. Comput. Syst., vol. 7, no. 3, pages 1–39, 2008. (Cited on page 69.)

[Singhoff 2004] F. Singhoff, J. Legrand, L. Nana and L. Marcé. *Cheddar: a flexible real time scheduling framework.* Ada Lett., vol. XXIV, pages 1–8, November 2004. (Cited on page 5.)

[Spuri 1996] M. Spuri and G. Buttazzo. *Scheduling aperiodic tasks in dynamic priority systems.* Real-Time Systems, vol. 10, pages 179–210, 1996. (Cited on page 13.)

[Storch 1996] M.F. Storch and J.W.-S. Liu. *DRTSS: a simulation framework for complex real-time systems.* Real-Time and Embedded Technology and Applications Symposium, IEEE, vol. 0, page 160, 1996. (Cited on page 5.)

[Thiele 2000] L. Thiele, S. Chakraborty and M. Naedele. *Real-time calculus for scheduling hard real-time systems.* In ISCAS, volume 4, pages 101–104, 2000. (Cited on pages 7, 69, 70, 72 and 81.)

[Thiele 2006] L. Thiele, E. Wandeler and N. Stoimenov. *Real-Time Interfaces for Composing Real-Time Systems.* In EMSOFT, pages 34–43, 2006. (Cited on pages 69, 76 and 79.)

[Tindell 1994a] Ken Tindell and John Clark. *Holistic schedulability analysis for distributed hard real-time systems.* Microprocessing and Microprogramming,

vol. 40, no. 2-3, pages 117 – 134, 1994. Parallel Processing in Embedded Real-time Systems. (Cited on page 6.)

[Tindell 1994b] K.W. Tindell and A. Burns. *Guaranteeing message latencies on Controller Area Network (CAN).* In Proceedings of 1st international CAN conference, pages 1–11, 1994. (Cited on page 63.)

[Tindell 1994c] K.W. Tindell, H. Hansson and A.J. Wellings. *Analysing real-time communications: Controller Area Network (CAN).* In Real-Time Systems Symposium, pages 259–263, Dec 1994. (Cited on pages 41 and 46.)

[Tindell 1995] K. Tindell, A. Burns and A.J. Wellings. *Calculating Controller Area Network (CAN) message response times.* Control Engineering Practice, vol. 3, no. 8, pages 1163 – 1169, 1995. (Cited on pages 32, 40, 42, 50, 53 and 59.)

[Wandeler 2005] Ernesto Wandeler and Lothar Thiele. *Real-time interfaces for interface-based design of real-time systems with fixed priority scheduling.* In EMSOFT '05: Proceedings of the 5th ACM international conference on Embedded software, pages 80–89, New York, NY, USA, 2005. ACM. (Cited on pages 69, 79 and 82.)

[Wandeler 2006a] Ernesto Wandeler and Lothar Thiele. *Interface-Based Design of Real-Time Systems with Hierarchical Scheduling.* In RTAS '06: Proceedings of the 12th IEEE Real-Time and Embedded Technology and Applications Symposium, pages 243–252, Washington, DC, USA, 2006. IEEE Computer Society. (Cited on pages 69 and 79.)

[Wandeler 2006b] Ernesto Wandeler and Lothar Thiele. *Real-Time Calculus (RTC) Toolbox.* http://www.mpa.ethz.ch/Rtctoolbox, 2006. (Cited on page 86.)

[Yen 1995] Ti-Yen Yen and W. Wolf. *Communication synthesis for distributed embedded systems.* In Computer-Aided Design, 1995. ICCAD-95. Digest of Technical Papers., 1995 IEEE/ACM International Conference on, pages 288–294, 1995. (Cited on page 6.)

[Yen 1998] Ti-Yen Yen and W. Wolf. *Performance estimation for real-time distributed embedded systems.* Parallel and Distributed Systems, IEEE Transactions on, vol. 9, no. 11, pages 1125–1136, 1998. (Cited on page 6.)

[Zeng 2009] Haibo Zeng, Marco Di Natale, Paolo Giusto and Alberto L. Sangiovanni-Vincentelli. *Stochastic Analysis of Distributed Real-time Automotive Systems.* IEEE Trans. Industrial Informatics, vol. 5, no. 4, pages 388–401, 2009. (Cited on page 69.)

[Zeng 2010] Haibo Zeng, M. Di Natale, P. Giusto and A. Sangiovanni-Vincentelli. *Using Statistical Methods to Compute the Probability Distribution of Message*

Response Time in Controller Area Network. IEEE Transactions on Industrial Informatics, vol. 6, no. 4, pages 678 –691, November 2010. (Cited on page 41.)

[Zhang 2008] Y. Zhang, D.K. Krecker, C. Gill, C. Lu and G.H. Thaker. *Practical Schedulability Analysis for Generalized Sporadic Tasks in Distributed Real-Time Systems*. Real-Time Systems, vol. 0, pages 223–232, July 2008. (Cited on page 13.)

Schedulability Analyses for the Design of Reliable and Cost-effective Automotive Embedded Systems

Abstract:
Keywords: